

COMPUTER PROGRAM SYSTEM STANDARDS

FACILITY FORM 001	N64-85463	
	(ACCESSION NUMBER)	(THRU)
	<i>130</i>	<i>None</i>
	(PAGES)	(CODE)
	(NASA CR OR TMX OR AD NUMBER)	(CATEGORY)



project
mercury

NEG.
MC 102

project
mercury

**PROGRAM SYSTEM
STANDARDS**

prepared for
National Aeronautics and Space Administration
Contract No. NAS 1-430

1 march 1961

International Business Machines Corporation

in association with

WESTERN ELECTRIC COMPANY, INC.

LIST OF EFFECTIVE PAGES

* The asterisk indicates pages changed, added or deleted by the current change.

CASE FILE COPY

TABLE OF CONTENTS

	<u>Page</u>
Section 1. INTRODUCTION	
1.1 SHARE OPERATING SYSTEM (SOS)	1-1
1.2 SHARE COMPILER-ASSEMBLER-TRANSLATOR (SCAT)	1-3
1.2.1 Compiler	1-3
1.2.2 Lister	1-4
1.2.3 Modify and Load	1-4
1.3 THE DEBUGGING SYSTEM	1-5
1.4 INPUT/OUTPUT SYSTEM	1-5
1.5 MONITOR—SUPERVISORY CONTROL	1-5
1.6 DIFFERENCES BETWEEN SOS SHARE AND SOS MERCURY . . .	1-5
Section 2. MODIFIED SOS SYSTEM	
2.1 COMPILER	2-1
2.1.1 SHARE Symbolic Language (SCAT)	2-1
2.1.2 Symbolic Input Format	2-2
2.1.3 Symbolic Language and Arithmetic Expressions	2-2
2.1.4 Evaluation of Variable Field Expressions	2-3
2.1.5 Special Characters	2-3
2.1.6 Classifications of Operation Codes	2-4
2.1.7 Machine Instructions	2-4

TABLE OF CONTENTS (Cont'd)

	<u>Page</u>
2.1.8 Pseudo-Instructions	2-4
2.1.9 Macro-Instructions	2-15
2.1.10 List Control Pseudo-Instructions	2-20
2.1.11 SCAT 709/7090 Machine Instructions	2-21
2.2 PROGRAM LISTINGS	2-22
2.2.1 Reference Systems	2-23
2.2.2 Sample Listing	2-24
2.3 MODIFY AND LOAD	2-25
2.3.1 Pseudo-Operations	2-26
2.4 DEBUGGING MACROS	2-37
2.4.1 Variable Fields	2-37
2.4.2 Information Macros	2-38
2.4.3 Modal Macros	2-39
2.4.4 Conditional Macros	2-40
2.4.5 Programming Examples of Debugging Macros	2-42
2.5 MONITOR	2-44
2.5.1 System Operation: Input Deck	2-45
2.5.2 Effect of Control Card	2-48
2.5.3 Specifications of the Data Sentence Program	2-50

Section 3. OTHER PROGRAMMING STANDARDS

3.1 MERCURY PROGRAM WRITEUP SPECIFICATIONS	3-1
3.2 FLOW CHARTING STANDARDS	3-3

TABLE OF CONTENTS (Cont'd)

	<u>Page</u>
3.3 PROGRAM CHECKS	3-5
3.4 SYMBOLS	3-6
 Section 4. MATHEMATICAL STANDARDS	
4.1 TERMINOLOGY	4-1
4.2 COORDINATE SYSTEMS AND CONVERSIONS	4-7
4.2.1 National Bureau of Standards Conversion Factors	4-19
4.2.2 Real Time Impact Prediction Coordinate Transformations	4-19
4.3 CONSTANTS	4-27
4.3.1 Constants (Alphabetical Order)	4-27
4.3.2 Constants (Numerical Order)	4-31
4.3.3 Octal Constants	4-34
4.4 TABLES	4-35
4.5 COMMUNICATION CELLS	4-47

LIST OF ILLUSTRATIONS

<u>Figure</u>	<u>Page</u>
4 - 1. Radar Coordinate System	4-11
4 - 2. Observational Framework, Inertial Coordinate System	4-12
4 - 3. Latitude Relationships	4-13
4 - 4. Longitude Relationships	4-14
4 - 5. Relationships Between Earth, Orbit and Capsule	4-15
4 - 6. Projection of Orbit on Celestial Sphere (unit vectors and angles displayed)	4-16
4 - 7. Projection of Orbit on Celestial Sphere (longitudes and anamolies displayed)	4-17
4 - 8. Xi, Eta Coordinates	4-18
4 - 9. Local Azusa (Mark I) Coordinate System	4-20
4 - 10. Local Radar Coordinate System	4-20
4 - 11. Inertial Coordinate System	4-22

LIST OF TABLES

<u>Table</u>	<u>Page</u>
1 - 1. Individual Files on the Mercury SOS Tape	1-7
1 - 2. Routines on the Project Mercury Library Tapes	1-9

Section 1

INTRODUCTION

A standard can be defined as a model or a set of criteria established by authority and/or general consent. The imposition of, or agreement upon, standards implies uniformity of results. The size and complexity of the Project Mercury Program System necessitated an immediate provision of standards.

Standards do not operate in a vacuum; they are applied, in the present instance, to a system—the Mercury Program System. The program system evolved from organized knowledge in existence at the beginning of the project—of primary interest was the SHARE Operating System (SOS). These modifications to SOS are discussed in detail in this volume, for they are basic to an understanding of the application of standards.

Standards were criteria for the entire programming effort. The use of such standards made it possible for programmers to develop their work independently of others and yet maintain a unified system. This system was the primary consideration in the delineation of programming guides.

The nature of the system required the utilization of two types of standards, programming standards and mathematical standards. The former provided guidelines for program writeups, flow charts and symbology; the latter aided in the organization of the presentation of terminology, coordinate systems and conversions, constants, tables and communication cells. These sections and their results (as displayed in the programming volumes) further illustrate the concept that standards are guides, not inflexible rules. These guides have been followed, generally, but there have been some deviations in detail.

1.1 SHARE OPERATING SYSTEM (SOS)

SHARE is a distribution agency, an organization formed by users of the IBM 709/7090 Data Processing Systems for the exchange of programming information and the mutual development of programming standards. The SHARE Operating System (SOS) is the primary programming system used with the IBM 709/7090 computers.

SOS was one of the existing programming systems surveyed by a Standards Committee for Project Mercury. The unique and stringent requirements for such a system, established by the original Mercury specification, called for the use of real time operation, internal and external interrupts, and special displays. The committee found that some programs would be almost impossible to code unless treated in a machine or pseudo-machine language. Therefore, a slightly

modified version of the SHARE Operating System was chosen as the basis for programming the Mercury project. (Differences between the SHARE and Mercury systems are discussed in Subsection 1.6 of this volume.)

To code each part of the project simultaneously, certain restrictions had to be placed on each programmer so the joint programming efforts could be compiled without difficulty. This necessitated the establishment of special features to guide the programmer in writing programs:

- a) A special notation for each section of the Mercury program.
- b) A special notation for communication between programmers.
- c) A special coding for machine hardware operation.
- d) The use of six-character symbols, communication cells and constants.

Certain constants such as the rotational speed of the earth had to be established; these values were determined by NASA and confirmed by Dr. Herget of the Cincinnati Observatory.

Mathematicians had to be consistent in their usage of Greek letters to designate the various units of space and time. These were agreed upon and are listed in the subsections titled "Symbols" and "Terminology" (Subsections 4.1 and 4.2 of this volume). There was a need for consistency in defining the coordinate systems to be used throughout all programs and in defining the conversion of local coordinate systems to spherical, and vice versa. Furthermore, because certain routines would be used to a great extent in Mercury system coding, a utility tape was developed to ensure consistency and ease of coding.

The use of SOS offers the following advantages:

- a) Relative ease in alteration of programs during the coding and testing stages.
- b) Control over the allocation of storage.
- c) Execution of various programs which are in different stages of development.
- d) Control over individual data (real time inputs), an absolute necessity.
- e) Incorporation of common library subroutines.

SOS also provides the advantages of symbolic assembly and eliminates the disadvantages of other assembly systems. Changes in symbolic form can be made in little more time than it takes to load binary punched cards. Debugging information can be listed in symbolic form rather than in actual language, as was previously required.

Other provisions of SOS include:

- a) The use of mnemonic operation codes (including a large group of pseudo-operations).
- b) Arbitrarily chosen location symbols.
- c) Relative and complex addressing.
- d) The definition of special-purpose macro-instructions for use in a given program.

Although SOS is actually an integrated system, it has, for convenience and easy reference, been divided into the following subsystems:

- a) The SHARE Compiler-Assembler-Translator (SCAT), subdivided into:
 - 1) Compiler
 - 2) Lister
 - 3) Modify and Load
- b) The Debugging system (program testing and correction), which includes the SNAP and SNAPTRAN programs.
- c) The Input/Output system.
- d) Monitor

1.2 SHARE COMPILER-ASSEMBLER-TRANSLATOR (SCAT)

This subsystem consists of three different parts: Compiler, Lister, and Modify and Load. These three parts together perform all the functions associated with symbolic assembly. In addition, SCAT produces symbolic listings, performs all the mechanics of incorporating modifications into a program, and loads programs for execution.

1.2.1 Compiler

The Compiler assembles the first part of a symbolic source program. This function consists of reading symbolic cards and translating the information contained in them into, and producing, a compact binary-coded-symbolic (squoze) form of the program. The squoze form of the program contains all the information, including remarks cards and comments from instruction cards, supplied in the source program.

The squoze deck produced by the Compiler may be used in either of two ways:

- a) It may be used with a symbolic deck and other squoze decks as input to subsequent Compiler passes, and incorporated with the symbolic deck to form one output squoze program. This feature makes it possible to write a program in parts and to debug each part before combining them.
- b) It may be used as input to Modify and Load, which completes assembly and loads the program for execution.

Another powerful tool of the Compiler is the macro-operation concept. The Compiler is built to recognize a large, fixed number of macro-operations. It also accepts and temporarily retains definitions of macro-operations given by the programmer. In either case, it generates and inserts into the program the sequence of machine words specified by any one of these macro-operations in a macro-instruction.

1.2.2 Lister

The SCAT Lister is actually a part of the Modify and Load program. However, since the Lister is used by the Compiler as well as by Modify and Load, and because knowledge of certain features of the listing produced by SCAT are required for an understanding of Modify and Load, the Lister is considered separately here.

The Lister provides the counterpart of program assembly listing. The listings produced include all the symbolic information, remarks and comments from the original source program deck as modified by subsequent changes. A machine language program is also generated by the Lister.

1.2.3 Modify and Load

Modify and Load completes the assembly of the input, incorporates symbolic modifications (if included with the input) and loads the program into storage for execution. Input to Modify and Load consists of a squoze program and, when necessary, symbolic cards which indicate changes to be made in the program.

Modify and Load also offers the following features:

- a) A new squoze program incorporating symbolic changes can be prepared when desired (a new listing of the program is also prepared).
- b) An absolute binary deck can be punched from a squoze program.
- c) A new listing of a program in squoze form can be prepared when required.

1.3 THE DEBUGGING SYSTEM

The Debugging system consists of a group of closed subroutines and their associated macro-instructions which may be written into a program at strategic points or included as program changes through Modify and Load. These subroutines supply the instructions necessary to print out symbolic information which aids in debugging.

1.4 INPUT/OUTPUT SYSTEM

The Input/Output system consists of a set of macro-instructions which generates in a program the instructions necessary for several types of input and output. These macro-instructions are general-purpose types and are intended to be interspersed with machine instructions, as necessary, to achieve special-purpose input/output for a given job.

1.5 MONITOR—SUPERVISORY CONTROL

Monitor is the control function of the SOS system. There is more than one Monitor system in SOS, but all perform essentially the same tasks. Input to the Monitor program consists of one or more job decks. A job deck is a program deck (symbolic, squeeze, or any combination of the two) with control cards which indicate the functions to be performed on the program, e.g., compile, list, load, etc. This deck is processed by SOS and is controlled while in process by the Monitor, as specified in the control cards in the job deck.

When a job deck is used as input, Monitor reads the control cards, determines the segment of SOS required for processing the deck, and loads the required part. Control is then transferred to the processor loaded by Monitor. That program then processes input until the end of the job deck is reached, a new control card is encountered or an error occurs. When the end of the deck is reached or a new control card is encountered, the Monitor is reloaded into storage and the process is repeated. If an error occurs, the Monitor prints a message indicating the error and, if possible, continues processing the job. If it is not possible for the Monitor to continue, it skips to the next job. (Monitor is discussed in detail in Section 2 of MC 107.)

1.6 DIFFERENCES BETWEEN SOS SHARE AND SOS MERCURY

A brief discussion of the SHARE Operating System was presented in Subsection 1.1. It is appropriate perhaps to examine some of the differences between SOS Mercury and the standard SOS (the term "standard" is almost meaningless, since there are many options in SHARE and, in all probability, few if any users have identical systems). An important point to mention here is the need for modification to a system (SOS) that seemed to fit the requirements for the Mercury Program System.

SOS is a dynamic programming system distributed by the SHARE agency to users of the IBM 709/7090 Data Processing Systems. To meet the changing needs of its members, SHARE constantly alters SOS; this particular factor is unnecessary, however, to the maintenance of a reliably operating Mercury Program System. For this reason, local changes were made.

The Mercury SOS System consists of:

- a) Several copies of the Mercury SOS Tape, which incorporate modifications through SHARE Distribution No. 30, and the local changes discussed later. (See Table 1-1 for the listing of files on the Mercury SOS Tape.)
- b) Two SOS Library Tapes and their duplicates. To realize a considerable conservation of core storage during compilation of the Mercury Program System, 13 of the 18 routines were recorded to provide common storage for the temporary results. The two tapes necessary are a tape on which each routine contains its own temporary storage (labeled "Regular SOS Library") and a tape on which the routines lack temporary storage (labeled "Mercury SOS Tape").
- c) The New York SOS System Tape (M 641) and duplicates—labeled "SHARE SOS Tape." This tape includes all sections of the system exactly as they have been developed in New York.
- d) The complete system is absolute (row) binary cards, used to write the Mercury SOS Tape, and another absolute (row) binary deck, used to write the SHARE SOS Tape. Each system contains approximately 3000 absolute binary cards.
- e) Each complete system (Mercury and SHARE) in column-binary squeeze cards with appropriate modifications. Each system contains approximately 10,000 squeeze cards.
- f) Five folders of listings for the sections of the SOS system: two folders with no modifications, one with New York modifications, one with all of the modifications, and one with the library routine listings.

The changes made to SHARE in developing the Mercury Program System are discussed under headings of the parts of the system: Monitor, Compiler, Modify and Load, Debug and Input/Output.

Monitor: The Mercury SOS Tape uses the New York 32K IBMonitor, with local modifications. SHARE intends to depart from this and adopt the Mock-Donald Monitor, but Mercury will retain the modified New York Monitor.

The local changes are:

- a) The Mercury Monitor initializes core storage above location 3000₁₀ to zeroes at the start of each job. SHARE initializes by inserting on STR instruction (operation code 1000₈) in each location above 3000₁₀.

TABLE 1-1. INDIVIDUAL FILES ON THE MERCURY SOS TAPE

File No.	File Name	Sequence Name	Section	Number of Records
1			Tape Loader	2
2	MN	MON	Monitor	13
3	M1	MLSUPR	Modify and Load	18
4	MO	MLMOD1		35
5	M3	MLMOD2		21
6	M7	MLPCH1		11
7	M7	MLPCH2		11
8	M7	MLPCH3		5
9	M7	MLPCH4		6
10	M7	MLPCH5		6
11	M4	MLASGN		8
12	M5	MLDCOD		13
13	M6	MLDERP		10
14	M8	MLLIST		43
15	M9	MLEROR		8
16	D1	SNP1	Debug	9
17	D3	DDE		9
18	D2	SNP2		20
19	C1	SCAT 1	Compiler	47
20	C2	SCAT 2		14
21	DA	DS 1	Input/Output	14
Files on the SHARE SOS Tape which have been omitted from the Mercury SOS Tape.				
	IN	INTRAN	Input/Output	
	OT	OUTRAN		
	TM	TM		

- b) For floating point overflow and underflow, Mercury prints the location of the instruction off-line, causing the overflow or underflow. Overflow sets bits 1 through 35 of the offending register (AC or MQ) to "1's" but the sign remains unchanged and the program continues. When an overflow occurs, SHARE dumps. For both SHARE and Mercury, underflow causes the offending register (AC or MQ) to be cleared, i.e., set to plus zero, and the program continues.
- c) A programmer may terminate his program by transferring to SYSTEM or SYSERR, without defining these symbols in his program, instead of returning to the SOS Monitor with a TRA 10 or 14, 110 or 114. System initializes for the next job; SYSERR gives an octal dump of core, from 3000₁₀ up, and then initializes for the next job.
- d) The table of names of the 18 routines on the Mercury Library Tape and the number of those routines appear in the Mercury Monitor. SHARE Monitor provides for these items but does not include them, since they are a function of each installation. (See Table 1-2 for the routines on Mercury Library Tapes.)
- e) The calling sequences to initialize the Intran and Outtran files were removed to allow room for the floating point overflow/underflow routine, because those files were omitted from the Mercury SOS Tape.
- f) To permit the changes mentioned above to be made without changing the correspondence of alter numbers and locations between the Mercury and SHARE systems, some instructions have been moved and some remarks have been inserted.

Compiler:

- a) A writeup was issued to allow for the possibility that it might be desirable in the future to incorporate some of the Mercury programmer macros into SOS as system macros (CORE, for example). (The Mercury SOS Tape does not include the Mercury programmer macros.) Core storage was made available to accomplish the insertions into the Compiler by removing certain instructions not used in Mercury (the instructions referring to data channels E, F, G and H, the magnetic drum and the cathode ray tube).
- b) A new SCAT instruction, PSLF (present sense lines) was created to permit the Mercury Monitor to activate the subchannels of the Data Communications Channel.
- c) When compiling squeeze cards with symbolic cards, and when using the control card SQZ or SZQRB, SHARE had previously incorrectly computed the checksum of each card to compare with what was punched on that card. This situation was corrected.

TABLE 1-2. ROUTINES ON THE PROJECT MERCURY LIBRARY TAPES**

Name (In order of appearance on the Library Tapes)	Description
UISICO	*Sine - Cosine
UIEXPE	*Exponential
UISQRT	*Square Root
UILOGE	Natural Logarithm
UIATAB	*Arc Tangent A/B
UIATNA	*Arc Tangent A
UIASCO	*Arc Sine - Arc Cosine
UITACO	*Tangent - Cotangent
UIFXPT	*Fix a Floating Point Number
UIFLPT	*Float a Fixed Point Number
U3DOTP	**"Dot" Product
U3XPRO	**"Cross" Product
U3MATM	Matrix Multiplication
UA1LSC	Convert XYZ Coordinates to RAE
U7INTP	Six-Point Lagrangian Interpolation
U3VMAG	*Generate Magnitude from Vector
U3VPRO	Vector "Triple Cross" Product
U3UNTV	*Generate Unit Vector from Vector

* Recoded for Library Tapes to store temporary results in COMMON.

** These routines are discussed in Section 3 of Volume MC 107.

Modify and Load:

- a) The PSLF instruction was added to the decode and lister files (see (b) under "Compiler").
- b) Deletions to allow room in storage for later additions of Mercury programmer macros to SOS were made in the same manner as those from the Compiler (see (a) under "Compiler").
- c) The decode and lister files were also altered to handle the compilation of squoze cards using SQZ or SQZRB.
- d) The Modify and Load supervisory controller and lister files were altered to permit the use of the system symbols SYSTEM and SYSERR.
- e) A logical error in the computation of the squoze card checksum during Compile and Punch Squoze was corrected by a change to the punch file.
- f) A logical error causing difficulty when altering out an END card with an alter number which was a multiple of 255 was corrected.
- g) It is now possible to alter out LBR cards at Modify and Load time; however, there is still no way to alter in a routine from the library tape.
- h) There is no longer any difficulty when a programmer's macro defines the symbolic location of the first instruction generated by the macro as a parameter of the macro.
- i) Another change to the modification file now permits the altering in or out of the TQO instruction.

Debug: There are no deviations from the SHARE debug section.

Input/Output: The Intran, Outtran and Transmission macros (dispatcher files) have been omitted completely from the Mercury SOS Tape, leaving the data sentence as the only I/O file retained. The only difference between the Mercury and SHARE data sentence files is that the system symbol SYSTRA is equated to 111 in Mercury and to 12 in SHARE. The Intran program is physically present on the tape, but only for use by the data sentence program.

Section 2

MODIFIED SOS SYSTEM

2.1 COMPILER

The Compiler in the SHARE Operating System performs three functions: translation, compilation and assembly. It processes the source program, written in symbolic language, and produces a tightly encoded binary deck.

Input to the Compiler can take the form of symbolic records and library routines; previously compiled programs can also be combined with subsequent symbolic programs as input. The output is a squoze deck of the compiled source program. The name "squoze" was adopted for the output deck and is meant to convey compactness. A squoze deck contains the source program coded in a compact form which retains the original symbolic information. It is this symbolic output that is loaded, modified and translated into actual machine language and executed by the Modify and Load section of the SOS system.

2.1.1 SHARE Symbolic Language (SCAT)

The mnemonic term SCAT is a contraction of SHARE Compiler, Assembler and Translator and is widely used as the name for the symbolic language in the SOS system. It is the logical extension of the SHARE symbolic language. The extensions which have evolved were dictated by the following general requirements:

- a) The capability to recognize all 709/7090 machine instructions and 709/7090 SHARE mnemonics.
- b) A requirement that IBM 704 programs be compatible with SCAT. The Compiler (CP) recognizes, with some modification, the 704 symbolic language (SAP). When a SAP pseudo-instruction is different from its SCAT equivalent, the Compiler converts it to a legitimate SCAT instruction. SCAT/SAP compatibility does not extend beyond the compiling phase, however. Modify and Load accepts only legitimate SCAT codes, treating all others as illegal.
- c) The incorporation of variable-length mnemonics, which provide a facility for expressing channel designation in a consistent way and provide a convenient means of specifying macro-instructions to be processed by the Compiler.
- d) For ease of key punching, it is desirable for the variable field to begin in the same column of every card, regardless of the length of the mnemonic.

2.1.2 Symbolic Input Format

The format of the symbolic instruction, with fields fixed at their maximum limits, is:

<u>Card Columns</u>	<u>Description</u>
1 - 6	Location field or blank
7	Blank
8 - 14	Operation code (including asterisk for indirect addressing)
15	Blank
16 - 72	Variable field and comments, which must be separated by a blank
73 - 80	Not used

In other words, the mnemonic operation code (beginning in card column 8) may be from one to six letters in length. At least one blank must follow the last letter; the number of blanks that may follow must be such that the length of the operation code plus the number of blanks is less than or equal to eight. If the variable field does not begin by column 16, it is assumed to be blank.

Four principal parts of a symbolic instruction are recognized: location symbol, operation code, variable field and comment field. The location symbol is a name for either a storage location or other expression associated with the instruction; the precise item named is dependent upon the operation specified. In all cases the operation determines the nature of the instruction and guides the interpretation of the various parts. The variable field is construed in a variety of ways as a function of the operation part of the instruction. In general, with the location symbol and operation, the variable field gives complete instruction specifications. The comment is not considered pertinent to the running of the program. It has the sole function of describing a remark intended to appear on a listing.

The order for the variable field of a 709/7090 symbolic instruction is address, tag, decrement. These subfields within the variable field are separated by commas. In all instructions it is possible to omit parts of the variable field. To omit only an interior part (the tag, for example) it is necessary to have two commas in adjacent positions because the first blank encountered in a variable field terminates that field. TXI A, 0, B and TXI A,,B result in the same word. Comments may begin after a blank, indicating the end of the variable field; however, for ease in key punching and to maintain uniformity, comments should begin in column 35. Comments may not begin before column 17.

2.1.3 Symbolic Language and Arithmetic Expressions

The basic units of the symbolic language are symbols, numbers, and operation codes. These units may be combined by punctuation marks, subject to certain rules, to yield expressions.

A symbol is a combination of from one to six Hollerith characters, at least one of which is non-numeric and none of which is either a +, -, *, ?, \$, =, comma or an imbedded blank. A blank is not considered a character in this case. A symbol is defined if and only if it appears in the location field of some instruction; otherwise, it is undefined. It is desirable to label a symbolic instruction with a location symbol only if it is necessary to refer to that instruction in the program. An absolute location symbol, i.e., one containing only numeric characters, is flagged as an error and is ignored. Leading zeroes are considered legitimate characters of a symbol.

A number is a combination of digits which may be decimal or octal, depending upon the operation code of the instruction in which it appears. An operation code may consist of from three to six alphabetic characters. An expression is a combination of symbols and integers separated by the following connectors or punctuation marks:

+	addition
-	subtraction
*	multiplication
/	division

(NOTE: These connectors have different meanings when used in the BOOL pseudo-operation. This operation is defined later.)

2.1.4 Evaluation of Variable Field Expressions

Constants in a variable field must be less than 2^{35} . They are considered decimal quantities unless the instruction is a Type D instruction. (NOTE: Examples of Type D instructions are: RIL 1, RIR 44, SIL 1 and LFT 2.) The constants of a Type D instruction are treated as octal values. Only simple expressions are permissible in the variable field of these Type D instructions, and the value is computed modulo 2^{18} . With all other instruction types, if the symbol referred to in a simple expression is octal (Boolean), the address and decrement fields are treated as 18-bit values and the tag is computed modulo 8. When not octal, the address and decrement fields are considered as 15-bit values and the tag is computed modulo 8.

2.1.5 Special Characters

The asterisk character, *, has five different meanings in SCAT, depending upon its context. As a punch in column 1 of a card it defines the card as a remark or comment card. If it is found immediately after an operation code, it specifies indirect addressing. As a connector in a variable field expression it connotes multiplication. As a Boolean operator it specifies intersection, e.g., the logical AND process. Finally, if it occurs immediately after another connector

or as the first character in a variable field, it must be recognized as a term. In this context an asterisk is interpreted as having the current value of the location counter.

The character, \$, may be preceded by a numerical, alphabetic or special character, or it may commence a term followed by five or fewer characters in an expression. These collocations cause SCAT to head the symbol with the given character rather than the current heading character. Reference from a headed region to an unheaded symbol is now made only by preceding the \$ with no heading character. Previously such referencing was also possible by preceding a \$ with zero.

2.1.6 Classification of Operation Codes

There are actually only two classifications of instructions: machine instructions and non-machine instructions. The latter type are collectively called pseudo-instructions. For purposes of this discussion, however, the pseudo-instructions are arbitrarily divided into three categories, one of which retains the generic name pseudo-instruction. In view of this arbitrary distinction, the Compiler, therefore, recognizes four classes of instructions:

- a) Machine instructions
- b) Pseudo-instructions
- c) Macro-instructions
- d) List control pseudo-instructions

2.1.7 Machine Instructions

A machine instruction (i.e., an instruction using a machine operation) always generates one 36-bit binary machine word in the object program. The rules for specifying the location field and the variable field of a machine instruction have been stated previously. The vocabulary of 709/7090 instructions and their SCAT mnemonics appear in Subsection 2.1.11. (For information concerning the operation of these instructions, refer to the 709/7090 Reference Manual.)

2.1.8 Pseudo-Instructions

Unlike machine instructions some pseudo-instructions may generate more than one machine word in the object program; others generate no words at all. The pseudo-operations of SOS have a variety of functions.

The rest of this section describes the pseudo-operations of the Compiler section of SOS (except for those which direct the Modify and Load program).

The mnemonics L and VF refer in the following paragraphs to location counter and variable field, respectively.

2.1.8.1 Assignment of Absolute Storage Locations—Origin (ORG)

The basic function of an assembly process is to assign absolute storage locations to machine instructions. There must be an address at which this assignment begins, however. In SCAT this value is furnished to the assembly program by the program being assembled via the ORG pseudo-instruction. ORG sets the location counter to the same integer value as that computed for its variable field. A location symbol associated with an ORG instruction is also assigned the computed value of the variable field.

		Address, Tag, Decrement
<u>Location</u>	<u>Operation</u>	
	ORG	100 ₁₀

In the example above, ORG assigns a value of 100₁₀ to the location counter. The location counter determines the storage location to which the subsequent instructions are assigned. The first instruction following the ORG card is assigned the location of the variable field value, modulo 2¹⁵, of the ORG card.

A symbol appearing in the variable field expression need not have been previously defined, i.e., need not have appeared in the location field (columns 1-6 of some previous instruction or pseudo-instruction). However, a symbol in the expression which is not eventually defined in the program renders the variable field of ORG non-computable.

If the program being assembled does not have an ORG pseudo-instruction, Modify and Load sets the ORG to the lowest location in memory not required by the SCAT system (3000₁₀). Subroutines assembled without ORG can be inserted into a job where needed, as long as they are prefaced by a SQZ control card.

2.1.8.2 Block Started by Symbol (BSS)

A BSS can occur anywhere in a program. This pseudo-instruction is used to reserve a block of storage whenever the program being assembled demands it. The block reserved is equal in length to the value of the variable field expression and extends from L to L + (VF - 1). The associated location symbol is given the value that L has when it encounters the BSS and corresponds, therefore, to the first word of the block reserved.

<u>Location Counter</u>	<u>Location</u>	<u>Operation</u>	Address, Tag, <u>Decrement</u>
250	A	BSS	200
450	B	XXX	XXX

In the example above, the BSS instruction reserves the 200 memory positions from locations 250 to 449, inclusive. The location symbol A is assigned to the value 250.

The rules for previous definition of symbols are the same as for the ORG pseudo-instruction.

2.1.8.3 Block Ended by Symbol (BES)

A BES may occur anywhere in a program. This pseudo-instruction is also used to reserve a block of storage at the direction of the program being assembled. In fact, a BES is the same as a BSS in every respect except for its result upon the associated location symbol. This symbol is given the value $L + VF$ and corresponds to the first word following the block reserved. Thus, whereas the associated location symbol in a BSS has the value of L , it is assigned the value of $L + VF$ in a BES instruction.

The rules for previous definition of symbols are the same as for the ORG pseudo-instruction.

The variable field of a BSS or BES may specify, as a tag, a code indicating the format of the data to be stored ultimately in the reserved block of storage. This specification is not required, but enables debugging programs to make a meaningful listing of such data. The codes are:

- F-- Floating point numbers
- X-- Fixed point numbers
- O-- Octal data
- H-- Hollerith (binary coded decimal data)
- S-- Symbolic instruction
- C-- I/O command
- V-- Variable Field Definition (VFD)

For example, if the programmer writes:

<u>Location</u>	<u>Operation</u>	<u>Variable Field</u>
Alpha	BSS	50, F

then he is, by using F, effectively saying to the Debugging system "the 50 cells in the block beginning at Alpha are to be interpreted as containing floating point numbers whenever I ask you later to give me the contents of any of these cells."

2.1.8.4 Transfer Card (TCD)

The purpose of this pseudo-instruction is to produce control information directing the loading program to execute a transfer of control from the loading program itself into the program being loaded. The transfer is made to the storage location represented by the value of the variable field expression of the TCD instruction.

There can be more than one TCD instruction and they can appear anywhere in the program.

If a TCD has an associated location symbol, the symbol is assigned the value that L has when it encounters the TCD instruction.

<u>Location Counter</u>	<u>Location</u>	<u>Operation</u>	<u>Address, Tag, Decrement</u>
200	A	TCD	2500

The instruction above sets A equal to 200; transfer of control is made to location 2500.

2.1.8.5 End (END)

Since, as explained in conjunction with the ORG pseudo-instruction, the computer must know where to start assigning absolute storage locations to machine instructions, it must also know when to stop this process. In SCAT, the termination of the assembly and loading operations is indicated by the END pseudo-instruction. It must appear in every program and must be the last instruction read during the assembly process.

As is the case with a TCD, the END instruction causes a transfer of control to be made to the storage location represented by the value of the variable field expression. The rules governing the associated location symbol, if any, are the same as TCD.

<u>Location Counter</u>	<u>Location</u>	<u>Operation</u>	<u>Address, Tag, Decrement</u>
800	A	END	1000

The instruction above sets A equal to 800; transfer of control is made to location 1000.

2.1.8.6 Equal (EQU)

The symbol appearing in columns 1-6 is assigned the integer value given by the expression appearing in the variable field.

The pseudo-operation EQU is used in those cases when the symbol appearing in columns 1-6 specifies a preset program parameter such as the number of items in a group, or any other quantity which is invariant with respect to the location of the program in storage. Note that if the symbol in columns 1-6 specifies the location of a piece of data or an instruction, the pseudo-instruction SYN should be used.

2.1.8.7 Synonym (SYN)

The symbol appearing in columns 1-6 is assigned the integer value given by the expression appearing in the variable field.

The pseudo-operation SYN is used in those cases when the symbol appearing in columns 1-6 specifies the location of a piece of data or other quantities whose values depend upon the location of the program in storage.

In SCAT language EQU and SYN may be used interchangeably, because in loading the subsequent squeeze deck by Modify and Load, the distinction is taken care of automatically. However, EQU and SYN have different effects if the binary object program is to be produced in a relocatable binary form. For the sake of clarity and use for later compilations, programmers should still make the distinction, using both the EQU and SYN pseudo-instructions.

2.1.8.8 Boolean (BOOL)

The BOOL pseudo-instruction is similar to EQU and SYN, for it defines a location symbol by equating it to the value of the single expression appearing in the variable field. All numbers in the variable field must be octal. The appearance of an 8 or 9 in the variable field indicates an error, and the computed value of the field is erroneous.

Computing the value of a Boolean variable field differs from computing the value of an ordinary expression because the Boolean punctuation marks specify logical rather than arithmetic operations, and the result is expressed modulo 2^{18} .

The punctuation marks, or operators, which may be used in this pseudo-instruction are:

<u>OPERATOR</u>	<u>MEANING</u>	
	<u>Algebra of Classes</u>	<u>709/7090</u>
+	Union	Inclusive OR
-	Symmetric difference	Exclusive OR
*	Intersection	AND
/	Complementation	Complementation

For example: SYMBL BOOL 505*317 results in an octal number of 105.

As with the EQU and SYN pseudo-instructions, the BOOL instruction must have a location symbol associated with it. The variable field of this instruction must be a single expression. Any division of the field into address, tag, decrement causes the tag and decrement parts to be ignored and results in an indication of possible error.

If the programmer is using the sense indicator register in his source program, he may often need to write Type D instructions, the 18-bit address part of which corresponds to the 18 leftmost or rightmost bits of this special register (see p. 60 and p. 51 of the 709 Reference Manual, A22 - 6501 - 1). If he cannot conveniently predetermine what particular sense indicator positions he would like to use, he might write, for example:

<u>Location</u>	<u>Operation</u>	<u>Variable Field</u>
:	:	:
	RIR	SENSX
:	:	:

Later, when he has decided that SENSX should be, say, the rightmost four positions (i.e., positions 32, 33, 34 and 35 of the sense indicator register) he can write:

<u>Location</u>	<u>Operation</u>	<u>Variable Field</u>
:	:	:
SENSX	BOOL	17
:	:	:

The 17 is interpreted as an octal number equivalent to (000 000 000 000 001 111)₂.

2.1.8.9 Heading (HEAD)

The HEAD pseudo-instruction provides a means of renaming symbols of fewer than six characters by inserting an additional character at the beginning of each symbol.

The variable field of a HEAD instruction must consist of only one character or a blank. Any other configuration results in an error indication and is ignored by the loading and assembly process.

The HEAD pseudo-instruction prefixes the heading character or blank to every location symbol and every variable field symbol of five or fewer characters encountered subsequent to itself and prior to the occurrence of another such instruction.

Location symbols and variable field symbols of six characters are not affected by the HEAD pseudo-instruction. This is significant, for it is through the use of six-character symbols and of the punctuation mark, \$, that reference from one headed field to another is possible.

A dollar sign appearing in a variable field is significant for the following reasons:

- a) An expression consisting of a single character, followed by a \$ and followed by a symbol of fewer than six characters is equivalent to the symbol headed by the initial character. For example, X\$A is equivalent to A headed by X. Such an expression is not affected by any HEAD pseudo-instruction.
- b) An expression consisting of a \$ followed by a symbol of fewer than six characters is equivalent to the symbol headed by a blank. Such an expression is not affected by any HEAD pseudo-instruction.

The following code illustrates the considerations mentioned above:

<u>Absolute Location</u>	<u>Symbolic Location</u>	<u>Code</u>	<u>Absolute Address</u>
0	A	CLA B	1
1	B	CLA A\$A HEAD A	2
2	A	CLA B	3
3	B	CLA \$A	0
4		CLA B\$A HEAD B	5
5	A	CLA A\$B	3
6		CLA \$X HEAD	7
7		CLA A	0
8		CLA B\$A	5
9		CLA COMMON HEAD C	11
10		CLA COMMON	11
11	COMMON	BSS, 1, F	11

Additional information:

- a) If no heading character is given, the Compiler heads with a blank. Heading can be discontinued by using HEAD with a blank variable field.
- b) Zero is a distinct heading character and indicates a heading.
- c) Reference to a headed symbol of five characters cannot be made by compounding a six-character symbol of the symbol and the heading

character. Thus, a reference in a variable field of ABCDE headed by X must be of the form X\$ABCDE and not XABCDE.

2.1.8.10 Decimal (DEC)

This pseudo-instruction is used to provide decimal data to the program being assembled. A single DEC instruction may specify more than one decimal number per card. Successive words are specified in the variable field and are separated by commas. The first blank encountered in the variable field terminates it. The data words generated by this instruction are assigned successively increasing storage locations; the location symbol, if present, is assigned the value of the storage location of the first word.

The sign of a number is indicated by a plus or a minus sign preceding the number, exponent, or binary scale factor. The absence of any punctuation implies a plus sign.

The variable field expression of a DEC instruction must be a numerical expression. The only characters admissible in such fields are commas, numerical constants, plus (+), minus (-), period (.), E and B.

Data generated by this pseudo-instruction is converted to one of three specified forms (binary integer, floating point binary number and fixed point binary number) according to the following rules;

- a) Binary integer (with the binary point at the right end of the word) if none of the characters, period (.), B, or E, appear in the numerical expression.
- b) Floating point binary number, if the characters, period (.) or E, or both, but not B, appear in the numerical expression. The appearance of E may be explicit or implicit.
 - 1) The decimal exponent to be used in the conversion is the number which immediately follows E. If E is not present, it may be implied by a signed number.
 - 2) The exponent is assumed to be zero if neither E nor a signed number appears.
 - 3) If the decimal point does not appear, it is assumed to be at the right end of the word.

The expressions +12.345, 12.345, 1.2345E1, 1.2345 +1, 1.2345E +1, 1234.5E-2, 1234.5-2 and 12345E-3 are all equivalent representations of the same floating point number, which is normalized following conversion.

c) Fixed point binary number, if the character B appears in the numerical expression:

- 1) The binary scale factor used in the conversion is the number immediately following B and may be positive, negative, or zero. (This factor is the count of binary positions between the left end and the binary point of the fixed point binary result.)
- 2) If the decimal point does not appear, it is assumed to be at the right end of the word.
- 3) The decimal exponent used in the conversion is the number immediately following E or, in the absence of E, implied by a signed number. If both B and E appear, the order of their appearance is irrelevant. For example, 1.2E1B4, 1.2B4E1, 1.2+1B4 and 1.2B4+1 are equivalent expressions.

Any word generated by a DEC pseudo-instruction which exceeds the limit of a machine cell results in a zero and an error is indicated.

In a DEC pseudo-instruction, a blank variable field, successive commas in the variable field, and a variable field ending in a comma all imply the generation of a zero.

2.1.8.11 Octal (OCT)

This pseudo-instruction is used to provide octal data to the program being assembled. A single OCT instruction may specify more than one octal number per card. Successive words are specified in the variable field and are separated by commas. The first blank encountered in the variable field terminates it. Data words generated by this instruction are assigned successively increasing storage locations and the location symbol, if present, is assigned the value of the storage location of the first word. (Note the similarity with the DEC pseudo-instruction, except for the kind of data generated.)

Octal numbers may be preceded by plus or minus signs; the absence of any sign implies a plus sign.

Octal numbers appearing in the variable field of OCT may consist of from one to twelve octal digits. The octal number may be signed if it is no greater in magnitude than 377777777777. If twelve digits appear, the following equivalences exist with respect to the sign and high-order digit: -0 = 4, -1 = 5, -2 = 6 and -3 = 7. If a sign appears with an octal number greater in magnitude than 377777777777, if more than twelve digits are written, or if any characters other than digits 0-7 appear in the variable field of this instruction, the conversion results in zero and an error is indicated.

In an OCT pseudo-instruction, blank variable field, successive commas in the variable field, and a variable field ending in a comma all imply the generation of a zero.

2.1.8.12 Binary Coded Information (BCI)

This pseudo-instruction is used to provide Hollerith data in standard binary coded decimal form to the program being assembled. The variable field of this instruction consists of one digit from 1-9, followed by a comma, followed by any characters (including the blank) which are acceptable to SCAT. Specified characters following the comma are packed together six to a 709/7090 word, and these words are assigned successively increasing storage locations. The number of words generated is specified by the digit preceding the comma. If a comma does not follow the first digit of the variable field, an error indication is given. Any location symbol associated with a BCI instruction is assigned the value of the storage location of the first word generated by the instruction. The use of another BCI card is required for more than nine words.

2.1.8.13 Library (LBR)

The LBR pseudo-instruction is used to extract a subroutine from a library tape and incorporate it into the program being assembled. The complete format is:

<u>Location</u>	<u>Operation</u>	<u>Address, Tag, Decrement</u>
SUBR	LBR	IDENT, U, CHANNEL AND TAPE NUMBER

If present, the location symbol is assigned to the first instruction in the library program, provided that the first instruction is not EQU, SYN or BOOL. If the first instruction already has a location symbol, it is equated to the location symbol of the LBR instruction.

IDENT and the Channel and Tape Number are only used to locate a subroutine in the tape library. The IDENT may be a symbol or an integer. If it is a zero or blank the location symbol is used as the identification. If the Channel and Tape Number is zero or blank it is assumed that the subroutine is on the SCAT library tape, and the location symbol is used as the label in this case.

The symbol U (unrelativized) indicates to SCAT that the library subroutine is not to be relativized. If the tag field contains any other symbol or is blank the program is to be relativized. Relativization is the process by which all addresses in the library subroutine are expressed relative to the first symbol in the library program, which is in effect a base point address.

The SAP pseudo-instruction LIB is changed by SCAT into an LBR and executed accordingly. However, the following conditions are assumed:

- a) The subroutine is on the system tape.
- b) The subroutine is relativized.
- c) The location symbol of LIB serves as the identification of the subroutine being called for by LBR. All addresses within a subroutine are expressed relative to a base point address.

2.1.8.14 Variable Field Definition (VFD)

This pseudo-instruction is used to specify the division of words in other than standard prefix, decrement, tag and address fields. The variable field consists of defining expressions, or subfields, which may specify three types of information: symbolic, octal, and/or Hollerith. These subfields within the variable field are of the following form:

$$n_1/E_1, On_2/E_2, Hn_3/E_3 \dots\dots$$

In the example above, n is a decimal constant indicating the number of bits to be occupied by the subfield; E is an ordinary variable field expression; H indicates a Hollerith subfield; and O indicates an octal subfield. All subfields are terminated by a comma or blank; these may not be included among the specified characters. If the given expression is longer than the designated n bits, the value of the subfield is taken modulo 2^n , i.e., the rightmost n bits are used. If it is shorter, the leftmost bits are filled in with blank characters in the case of a Hollerith subfield and with zeros for all other types of subfields.

The first subfield specified begins at the leftmost part of the first word generated. If a location symbol appears it is equated to the location of this word. The next subfield begins to the right of the previously defined subfield. If a subfield extends beyond the end of a word, it is continued from the left end of the next word.

There is no limit to the number of subfields which may be specified by this pseudo-instruction; the length of any subfield cannot exceed 63 bits, however.

All subfields give the actual expression and not the location of the expression. All expressions are computed modulo the length of the subfield rather than in the usual manner.

The expressions of a VFD variable field may be either ordinary or Boolean, or both, but they cannot both be in the same subfield.

Unless prefixed by O , the numbers in the variable field expression are to the base 10 even when they occur with Boolean symbols.

2.1.8.15 Et Cetera (ETC)

This pseudo-instruction is only used to extend the variable field of the previous instruction. The variable field of the previous instruction must be terminated by a comma. If the comma has significance within the field, the break must be made at an insignificant comma. If the previous variable field does not terminate with a comma, a comma is assumed and an error is indicated. In any event, the variable field of an ETC pseudo-instruction is considered an extension of the variable field of the previous instruction, commencing at a comma and thus with a complete expression.

An ETC pseudo-instruction may not have a location symbol associated with it. It may, however, follow any instructions possessing a variable field. The following points about ETC should be clearly understood:

- a) If a comma has significance within a field which is being extended by an ETC instruction, the break must occur at a comma which separates fields, i.e., the comma signalling the ETC must not be introduced within an expression.
- b) The variable field of ETC does not begin with a comma. In fact, it does not differ from any other variable field. In the preliminary description of SCAT, it is stated that "the variable field of an ETC pseudo-instruction is considered an extension of the variable field of the previous instruction, commencing at a comma and thus with a complete expression." This is true but could be misleading. The critical word is at--the expression commences at a comma, but not with a comma.
- c) An ETC may follow only a VFD pseudo-instruction, the MACRO pseudo-instruction or any operation code calling for the generation of a system or programmer macro, and nothing else.

2.1.8.16 Remarks (*)

The pseudo-instruction asterisk, *, indicated in column 1 of a card, is used to enter into the program being assembled commentary material which is to appear on a listing. The remaining 71 positions of the symbolic card may be used as a comment field.

This pseudo-instruction has no location field (the asterisk is not recognized as such), operation code field or variable field. It has no effect upon the assembly process.

2.1.9 Macro-Instructions

A macro-instruction generates a word or a sequence of words. Parameters required by the macro subroutine must appear in the variable field of the

macro-instruction. These parameters are incorporated into the word or sequence of words generated by the macro-instruction during the compilation of the object program rather than at the execution time of the object program.

There are two types of macro-instructions in SCAT: system and programmer. The difference between system and programmer macro-instructions is that the former are provided for in the Compiler, and the latter are innovated in the source program.

2.1.9.1 System Macro-Instructions

The generation of a system macro-instruction is called for whenever its code name appears in the operation code field. The variable field specifies the parameters to be used in the generated sequence of words. Any location symbol associated with the macro-operation is assigned as the location symbol of the first of the generated words.

At present there are two macro-instructions which have been incorporated into the Compiler: BEGIN and RETURN. It is assumed that many such macro-instructions will be available in the Compiler and that others will be added by installations to handle special jobs.

BEGIN K, T, I, E: The BEGIN macro-instruction generates the basic subroutine linkage recommended by the SHARE Operating System Committee. The parameters K, T, I and E are significant for the following:

K -- Location of the normal return relative to the TSX. The exit transfer is TRA K, 4.

T -- Specification of the index registers to be saved. It is recommended as a debugging aid that index register 4 always be saved.

I -- If I is 1, the sense indicators are to be saved and restored; if I=0, or blank, they are not to be saved and restored.

E -- Specify whether to save and restore a cell to indicate what channel traps should be enabled.

The number of resulting instructions equals $2X + 3I + 2$, where X is the number of index registers specified by T and I is as defined above.

The maximum and minimum sequences are given below to the right of the corresponding macro-instructions.

Maximum Sequence:

SR	BEGIN	2, 7, 1	SR	TXL	*+7
				AXT	0,4
				AXT	0,2
				AXT	0,1
				LDI	*+2
				TRA	2,4
				PZE	
				STI	*-1
				SXA	*-5,1
				SXA	*-7,2
				SXA	*-9,4

Minimum Sequence:

SR	BEGIN	2, 4	SR	TXL	*+3
				AXT	0,4
				TRA	2,4
				SXA	*-2,4

A RETURN SR, n: This macro-instruction specifies the error code and generates the instructions necessary for the normal and error exits from the routine. If present, A is the location of the first generated instruction; SR identifies the subroutine. This identification is necessary since RETURN need not refer to the most recent BEGIN macro-instruction. The error code n is stored in the decrement of the first generated instruction of the associated BEGIN.

If no error return procedure is desired, n is zero or blank. In this case, one instruction results.

TRA SR+1

If n is specified, the following sequence is generated:

AXT	n, 4
SXD	SR, 4
LXA	SR+1, 4
TXI	SR+2, 4, 1

The use of the system macro-instructions is illustrated below:

	<u>Source</u>	<u>Program</u>		<u>Object</u>	<u>Program</u>
SR	BEGIN	2, 7, 1	SR	TXL	*+7
	TPL	SR2		AXT	0, 4
SR1	RETURN	SR, 1		AXT	0, 2
SR2	DVP	X		AXT	0, 1

	<u>Source</u>	<u>Program</u>	<u>Object</u>	<u>Program</u>
SR3	STQ	Y	LDI	*+2
	RETURN	SR	TRA	2, 4
			PZE	
			STI	* -1
			SXA	* -5, 1
			SXA	* -7, 2
			SXA	* -9, 4
			TPL	SR2
			SR1 AXT	1, 4
			SXD	SR, 4
			LXA	SR+1, 4
			TXI	SR+2, 4, 1
			SR2 DVP	X
			STQ	Y
			SR3 TRA	SR+1

2.1.9.2 Programmer Macro-Instructions

In addition to system macro-instructions, the Compiler processes macro-instructions defined by the programmer for use in the program being compiled. The definition is introduced to the Compiler by the MACRO pseudo-instruction, which must have the code name of the programmer macro in its location symbol field and the code MACRO in its operation field. The location symbol must be from one to five characters in length, must be completely alphabetic and must not be the code name of a machine operation, a pseudo-operation or a system macro-operation. If the given code name is that of a previously defined programmer macro-instruction, the new definition replaces the former one.

The MACRO card lists in its variable field the parameters to appear in the defining example. All of these parameters must be non-constant. The variable field may be extended by ETC cards. However, the maximum number of parameters which can be specified by a MACRO pseudo-instruction and its associated ETC cards is 32. They are separated by commas.

The instructions which constitute the defining example follow the MACRO card in a sequence terminated by an END card. A defining instruction may have in its variable field any valid combination of symbols and connectors. All location symbols are variable field symbols of the defining example and must have appeared in the parameter list of the MACRO card.

Although the example used to illustrate the technique of writing macro-instructions shows all the variable field symbols as appearing in the parameter list, it is not necessary that such symbols be included among the parameters. It is true, however, that all location symbols must appear elsewhere in the program.

If symbols appear both in the parameter list and elsewhere in the program, preference is given to their definitions in the parameter list in attempting to define a programmer macro-instruction.

The following illustrates a MACRO pseudo-instruction and its defining example:

POLY	MACRO	COEFF, INVAR, DPVAR, DEG
	ETC	T, Z
	AXT	DEG, T
	LDQ	COEFF
DPVAR	FMP	A\$INVAR
	Z	COEFF+DEG+1, T
	XCA	
	TIX	DPVAR, T, 1
	END	

The location symbol of the MACRO pseudo-instruction becomes the operation code of the defined programmer macro-instruction. The number of instructions generated by a programmer macro-instruction is always the same as the number in the defining example. For example, the symbol POLY defined above could be used to form the macro-instruction:

POLY C1+10, X, FX, 3, 4, FAD

which would then generate the following sequences, or skeleton, in accordance with the pattern of the defining example:

	AXT	3, 4
	LDQ	C1 + 10
FX	FMP	A\$X
	FAD	C1 + 14, 4
	XCA	
	TIX	FX, 4, 1

In the coding example, the first two instructions of the defining example are:

POLY	MACRO	COEFF, INVAR, DPVAR, DEG
	ETC	T, Z

The entire example is correct as shown. It is desirable, however, to be very explicit about the following:

A parameter used in the defining example may not be the mnemonic for any instruction.

As the example shows, it is permissible to have one of the parameters represent an operation code in the manner in which Z stands for FAD. This

means that an operation code may be included among the parameters of a defined macro-instruction, as the included example illustrates:

POLY C1 + 10, X, FX, 3, 4, FAD

The restriction mentioned here applies only to the parameter list of the defining example.

A system macro can occur in the definition of a programmer macro; a programmer macro cannot occur in the definition of a programmer macro.

Note that the parameters of the defined macro-instruction may be symbolic or absolute, that they have a one-to-one correspondence with the dummy parameters of the MACRO pseudo-instruction, and that they have replaced them in the generated skeleton. Symbols which are to appear in the variable fields of the generated instructions may appear elsewhere in the source program. However, symbols to appear in the location fields of the generated instructions must not appear elsewhere in the program. This would result in multiple definition of the symbols.

2.1.9.3 Properties of Both System and Programmer Macro-Instructions

- a) A location symbol is identified with the first instruction generated.
- b) The variable field may consist of expressions and simple symbols. Any expression which ultimately appears as a divisor of a fraction in a variable field may have only one symbol.
- c) The variable field may be extended by ETC cards.

2.1.10 List Control Pseudo-Instructions

The Compiler provides the following as a listing: symbolic program with comments and alter and relative numbers (to be described in the Modify and Load section); page heading, page number and date on each page; an optional octal or decimal absolute program; error tables containing duplicated symbols, undefined symbols, and the total number of error-flagged instructions; and an optional symbol table which gives the symbol and page number. The list may be used in finding symbols in the listing when no absolute program is printed.

The following list control pseudo-instructions are provided to edit the listing of any program.

LIST: The LIST pseudo-instruction causes printing in the normal mode--all cards are listed without printing in detail, i.e., without printing of words generated by pseudo-instructions (OCT,VFD,DEC,LBR and BCI) or by macro-instructions.

UNLIST: An UNLIST instruction causes complete suspension of printing until a LIST instruction is encountered.

DETAIL: If the instruction DETAIL (with a blank variable field) is encountered, any printing which is currently in progress continues with complete detail, i.e., the machine words generated by macro-instructions (system and programmer macros), LBR, DEC, OCT, BCI and VFD instructions, are printed. The effect of a DETAIL instruction is nullified when and only when a TITLE, LIST or UNLIST instruction is encountered.

TITLE: A TITLE instruction causes any printing which is currently in progress to be continued in the normal mode (i.e., without any detail) until a subsequent DETAIL instruction or, of course, UNLIST instruction is encountered. If printing is already in progress in the normal mode, or if no printing is in progress at all, a TITLE instruction has no effect.

2.1.11 SCAT 709/7090 Machine Instructions

Included in the list of instructions (although they are not actual machine instructions) are those operation codes which may be used to assign arbitrary values to the prefix and sign of calling sequence words. They are listed as a group below:

<u>Alphabetic Code</u>	<u>Name</u>	<u>Octal Code</u>
MZE	Minus zero	-0000
MON	Minus one	-1000
MTW	Minus two	-2000
MTH	Minus three	-3000
PZE	Plus zero	+0000
PON	Plus one	+1000
PTW	Plus two	+2000
PTH	Plus three	+3000
FOR	Four	-0000
FVE	Five	-1000
SIX	Six	-2000
SVN	Seven	-3000

Instructions are listed below with information concerning address (A), tag (T), decrement (D) and indirect addressing (I). Codes appearing under the various headings have the following significance:

N-- This entry under the columns A, T, D and I indicates that the corresponding instruction should not have an address, tag, decrement or indirect address, respectively. A zero in the address, tag or decrement does not violate this restriction. If the prescribed field is specified, it is processed as given and an error is noted.

- Y-- This entry under a column heading indicates that the specified parts of the corresponding instruction should occur. If the field is to be provided by the program, a zero should be used.
- O-- This heading under column A indicates that the address field must be an octal number or Boolean symbol.
- I-- This entry under column T indicates that the tag field, if specified, must be a 1 or an expression with an equivalence of 1. No other non-zero tag is permitted.
- C-- There are six instructions (CAQ,CRQ,CVR,VDH,VDP,VLM) which use the decrement field as a count. C appears under column D of these instructions to indicate that the count is required.

2.2 PROGRAM LISTINGS

This subsection describes the form of program listings produced by the SOS system. The material is included here in preparation for the discussion of the Modify and Load pseudo-operation presented in Subsection 2.3, since references to information in the program listings are necessary in that subsection.

The purpose of the SOS listing facilities is to provide means of obtaining necessary information when making program modifications. Listings produced by SOS are made in symbolic form, since this is the most useful method for determining necessary changes.

Symbolic listings of a squeeze deck reproduce, with some exceptions, the symbolic source deck program, including modifications incorporated by the punching of a new squeeze deck. The exceptions which are never reproduced are:

- a) Invalid operation codes, which are replaced in the listing by ///.
- b) Invalid symbols, such as those longer than six characters, which are replaced by //////.
- c) The shortened forms of extended operation codes, which are changed and listed in their extended forms, e.g., the instruction WRS 1169 is listed as WTBB 1.

In addition, words generated by the BCI, DEC, LBR and OCT instructions or by macro-instructions are not normally listed in detail. Instead, only a title line and the first word generated by these instructions are printed. These may be listed in detail, however, if the pseudo-op DETAIL is used as previously defined.

When a squeeze deck is listed, the comments are aligned with the first comment in the program and therefore may not be lined up exactly as in the source deck listing.

Symbolic listings show the job title, page number and date in the upper right hand corner of each page and are followed by 50 lines of printing. The listing itself consists of several parts.

The symbolic instructions for the program are listed. In addition, octal equivalents are normally given. These instructions are given numbers from two reference systems (i.e., relative and alter numbers) which are assigned as described below.

Appearing next, at the option of the user, is a listing of all defined symbols used in the program. These symbols appear five on each line, in alphabetic order. Multiply-defined symbols appear at the end of the table, with the numbers of all the pages on which they appear.

2.2.1 Reference Systems

The two numbering systems previously mentioned (relative and alter numbering) are used to refer to words in a program. These numbers are assigned initially by the Compiler and are changed, if necessary, by Modify and Load only when a new squeeze deck is punched.

2.2.1.1 Relative Numbering

A relative number is an integer used to indicate the position of a machine word, not assigned a location symbol, relative to the preceding word in the program with which a location symbol is associated. The positions thus indicated are the relative positions of instructions the last time a squeeze deck was punched.

Since relative numbers in a sense indicate storage locations occupied by machine words, they are assigned only to those instructions which, when loaded for execution, occupy locations. Thus, relative numbers are never assigned to principal pseudo-instructions (BES, BOOL, BSS, END, EQU, HEAD, ORG, SYN, TCD), generative pseudo-instructions (BCI, DEC, DUP, LBR, OCT) or programmer macro-instruction definitions.

Relative numbering begins when the first location symbol of a program is encountered. The word associated with this symbol is numbered 0 (although not shown on listings) and the next word is numbered +1. Numbering continues until either another word with a location symbol or an instruction with a principal pseudo-operation is encountered. When a new symbol is encountered the process is started again. If, however, relative numbering is suspended by one of the pseudo-operations, it is not reinitiated until a new symbol is encountered. Words for which a positive relative number cannot be computed are given a negative relative number, i.e., a number relative to a succeeding symbol, if that can be computed. If neither can be computed no relative number is shown.

Although only one relative number is shown on the listing for a given word, there exist, in general, many other equivalent relative numbers, both positive and negative, any one of which may be used when referring to that word. For example, in the following list, the word numbered +1, relative to the symbol MASK, has the equivalent number +7, relative to RESTOR, or -1, relative to WRITE, etc.

82	RESTOR	AXT	**0, 1	RESTORE
83	+1	AXT	**0, 2	INDEX REGISTERS
84	+2	AXT	**0, 4	CONTENTS
85	+3	AXT	2, 4	RETURN
86	+4	SLN	1	TURN SENSE LIGHT 1 ON
87	+5	TRA	PRINT	
88	MASK	OCT	373737373737	
89	+1	OCT	377737773777	
90	WRITE	PZE	WKAREA,, 24	
91	IMAGE	BSS	NUMBER, 0	
92	NUMBER	EQU	24	
93	ZERO	EQU	0	
94	TSTBIT	PZE		STORAGE FOR TEST BIT
95		END	PRCOMM	

There is no number for a word relative to a symbol which is separated from that word by a principal pseudo-operation. For example, in the listing the words preceding the BSS with the location symbol IMAGE have no numbers relative to the symbol TSTBIT.

2.2.1.2 Alter Numbering

Alter numbers are numbers for the symbolic cards in a source program deck and are assigned to all cards except:

- Those which contain ETC and MACRO instructions.
- Those which define programmer macro-instructions.
- The Modify and Load pseudo-instructions.

Generative pseudo-instructions (such as BCI) and programmer macro-instructions are assigned alter numbers. The words generated by the instructions are not assigned numbers.

2.2.2 Sample Listing

The following sample presents the data found on an SOS symbolic listing:

- Storage locations in octal
- Octal equivalent of each instruction

- c) Alter numbers
- d) Symbolic locations
- e) Relative numbers
- f) Operation codes
- g) Variable field (containing the address, tag and decrement portions and/or a comment section)**

a	b	c	d	e	f	g
						TRI FUN HLS 00/00/00 Page 1
		1*				TRIG FUNCTION Problem
		2*				HOMER SNIDER
		3*				JOB VGPP, SIN, COS
		4			ORG	15000
35230	0 77400 1 00132	5	XA		AXT	90, 1 Generate
35231	0 50000 0 35602	6	XA1		CLA	ZERO FIXED
35232	0 40000 0 35736	7		+ 1	ADD	ONEX Point
35233	0 60100 1 35736	8	XA2		STO	FIXED-91, 1 Numbers
35234	2 00001 1 35232	9	XA3		TIX	*-2, 1, 1 0 to 90
35235	0 77400 1 00022	10	XA4		AXT	18, 1 Generate
35236	0 50000 0 35737	11	XA5		CLA	ONEF Floating
35237	0 30000 0 35737	12		+ 1	FAD	ONEF Point
35240	0 60100 1 36010	13	XA6		STO	Float-18, 1 Numbers
35241	2 00001 1 35237	14	XA7		TIX	*-2, 1, 1 2 to 19
35242	0 50000 0 35737	15	XA8		CLA	ONEF Float one.
35243	0 60100 0 35740	16		+ 1	STO	E
35244	0 77400 2 01166	17		+ 2	AXT	630, 2
35245	0 77400 1 00266	18		+ 3	AXT	182, 1
					CORE	FIXED, X, 0, 0
35246	0 62500 0	19	CORE	1	STL	2169

**The variable field section must start in column 16. The comment section must be separated from the end of a preceding variable field by at least one blank. In no case can it start to the left of column 17.

2.3 MODIFY AND LOAD

The input to the SOS Compiler is a symbolic source program from which is produced a compact binary coded symbolic (squoze) program deck containing all of the information supplied in the source program, including remarks cards and comments from instruction cards. Squoze decks produced by the Compiler may

be used with symbolic decks as input to subsequent Compiler passes to produce one squoze output deck. Thus, a program can be written in parts and each part debugged before all are combined.

Squoze decks produced by the Compiler are also used as input to Modify and Load. Since all symbolic information is available to Modify and Load, three major advantages over previous assembly systems are provided:

- a) Changes can be specified in symbolic form for incorporation into the program by Modify and Load.
- b) Symbolic changes do not require the source deck to be reprocessed by the Compiler.
- c) Symbolic information is available and may be retained for printing during debugging runs, thus making debugging easier.

The main functions performed by Modify and Load are:

- a) Modification of a squoze program on the basis of symbolic information supplied with the squoze deck.
- b) Loading the modified version of a program into storage in preparation for execution of the program.

In addition to the above, Modify and Load also offers the following features:

- a) When desirable, a new squoze deck incorporating symbolic modifications may be prepared. (A new squoze deck is automatically prepared when a modification affects a heading card.) It is desirable generally to exercise this option when the number of modification cards is approximately equal to the number of cards in the squoze deck.
- b) A symbolic listing of a program can be prepared from a squoze deck which includes no modifications. (A new symbolic listing is automatically prepared when a new squoze deck is punched.)
- c) An absolute binary version of a program may be obtained from a squoze deck. Although this option is available to the user, little benefit is derived by exercising the option until a program has been completely debugged, because the debugging and modification features of SOS can only be used with squoze program decks.

2.3.1 Pseudo-Operations

The SCAT language includes five pseudo-operations by which changes may be made to a program at Modify and Loadtime. The use and effect of these pseudo-operations are described below.

To accomplish modifications, the modification instructions and any words to be inserted into a program are punched in symbolic form and used as input with the squeeze deck. The changes indicated in these cards are made in the program before it is loaded into storage but do not affect the squeeze deck until a new deck is punched. At that time, the changes are physically incorporated into the new squeeze deck.

The effects of the modification pseudo-operations when loading a program into storage and when preparing a new squeeze deck are equivalent to and could be accomplished by making the required changes in the original symbolic source program, reprocessing with the Compiler and then loading the new squeeze deck. In the discussion that follows, only the effects which the pseudo-operations have on the squeeze deck are indicated.

Throughout the discussion each change is indicated as though it were the only one affecting the program, regardless of the actual number. That is, all changes must be indicated in terms of the current deck and the associated listing.

2.3.1.1 CHANGE

The CHANGE pseudo-operation can be used to delete words from a program, to insert additional words into a program, or to do both, depending on the form of the instruction. When CHANGE is used, modifications are specified in terms of relative numbers.

CHANGE instructions may be used to delete or insert words with which location symbols are associated, in which case the location symbol is also inserted or deleted. When a word which has a location symbol is deleted, the symbol is deleted from the dictionary and may, therefore, be used subsequently as a location symbol for another word. No location is required with CHANGE; if one is present it is ignored.

Two forms of the CHANGE instruction are permissible. The first is:

Location			Operation			Address, Tag, Decrement/Count
1	2	6	7	8	14	15 16
				CHANGE		A + n, B + m

A + n and B + m represent relative expressions, i.e., A and B are symbols and m and n are integers which may be positive, negative or zero.

This form indicates that all words in location A + n to B + m, inclusive, are to be deleted from the program. If, in addition, symbolic instruction cards immediately follow an instruction in this form, the instruction also indicates that the words in the symbolic cards are to be inserted beginning with location A + n. Since insertions are made as in an assembly, the words following B + m are

automatically adjusted and the number of insertions and deletions need not be equal.

When any, but not all, of the words generated by either BCI,DEC,LBR,MACRO or OCT are deleted by a CHANGE, each of the subfields remaining from the original instruction is carried as a separate word and is assigned a separate alter number. In the listing, however, only the absolute word and relative and alter numbers are shown. No symbolic information is shown in the operation, variable and comments fields. In all other changes to which a CHANGE can apply, the comments associated with deleted words are deleted from the squeeze deck; remarks cards falling within the range of deletion by a CHANGE are not deleted from the program.

When a CHANGE instruction of the form shown above affects a headed area, it must be written:

	Location			Operation			Address, Tag, Decrement/Count
1	2	6	7	8	14	15	16
				CHANGE			H\$A, H\$B +m

H represents a heading character.

The second form permitted is:

	Location			Operation			Address, Tag, Decrement/Count
1	2	6	7	8	14	15	16
				CHANGE			A + n

A is a symbol and n is an integer which may be positive, negative or zero.

This form of a CHANGE instruction indicates that the symbolic instruction cards which immediately follow it are to be inserted between the words in location A + n and A + n + 1. No deletions are caused by this form. If no symbolic cards follow an instruction in this form the instruction is ignored.

When a generative pseudo-operation is inserted into a program by means of a CHANGE instruction, the individual terms are not assigned separate alter numbers.

When insertions are to be made in a headed area, the second form of the CHANGE instruction is written:

	Location		Operation		Address, Tag, Decrement/Count
1	2 6	7	8 14	15	16
			CHANGE		K\$A+n

K represents a heading character and A+n is as previously described.

In the following list of restrictions all statements are made in terms of the headed forms of CHANGE. These restrictions can be applied to the unheaded forms by considering an unheaded symbol to be headed by the character blank.

Restrictions:

- a) In a CHANGE instruction of the first form, H\$A+n must be either less than or equal to H\$B+m; otherwise, the CHANGE and the symbolic cards following it are ignored.
- b) No principal pseudo-operation (BES,BOOL,BSS,END,EQU,HEAD,ORG,SYN,TCD) may appear within the range of the symbols A to B+m.
- c) No principal pseudo-instruction, listing pseudo-instruction or remarks card may appear as an insertion by means of a CHANGE. Any insertion which violates this restriction is ignored.
- d) Remarks cards and listing pseudo-operations cannot be deleted by a CHANGE. When remarks cards or pseudo-operations appear between H\$A+n and H\$B+m, inclusive, they are not affected by the CHANGE.
- e) No CHANGE instruction should specify the deletion of only part of the words generated by a VFD pseudo-operation.
- f) If a programmer macro-instruction is inserted by means of a CHANGE, the definition must also be included with the group of modifications. This does not mean that the definition must be included with the same CHANGE that is to insert the macro-instruction. Instead, it may be included by an ALTER or by another CHANGE. The definition may also be placed in front of the group of modifications and need not be preceded by a Modify and Load pseudo-operation.
- g) A modification by a CHANGE instruction must not overlap another modification by an ALTER (see below) or by a CHANGE.

Example 1: Assume that in the following listing the instructions with the alter numbers 79 and 80 are indicated to be in error.

```

78          1      TRA      CLEAR+4      RETURN
79      EXIT          AXT          , 1
80          1      ///          1      IF SENSE LIGHT 1 IS ON
81*                                     DO NOT RESTORE INDEX
                                           REGISTER 1

```

To remove the error indication by means of a CHANGE, the following instructions are necessary:

Location		Operation		Address, Tag, Decrement/Count	
1	2 6	7	8 14	15	16
EXIT →			CHANGE AXT SLT		EXIT, EXIT+1 **0, 1 1

(NOTE: **0 was arbitrarily selected to indicate modified addresses.)

Assuming there are no modifications which affected the alter numbering of previous instructions in the listing, the instructions corrected would appear in a listing of the modified deck as:

```

78          +1      TRA      CLEAR+ 4      RETURN
79      EXIT          AXT      **0, 1
80          +1      SLT      1
81*                                     DO NOT RESTORE IR 1

```

(The octal absolute has been omitted for the sake of clarity; however, the absolute equivalents would also be changed.)

Example 2: Assume that the instruction SLT 1 is to be inserted following the instruction in the list below, which has an alter number 9, without deleting any instructions.

```

6  PRCOMM  CLA      1, 4      GET PRINTER CONTROL WORD
7      +1   TDL      * 2
8      +2   WPDA
9      +3   WPDA      DOUBLE SPACE PRINTED IF
10     +4   SXA      RESTOR, 1  CONTROL NEGATIVE, SINGLE IF+
11     +5   SXA      RESTOR+1, 2 SAVE INDEX
                                           REGISTER

```

The required modification cards are:

The required modification cards are:

1	Location 2 6	7	Operation 8 14	15	Address, Tag, Decrement/Count 16
			CHANGE SLT		PRCOMM + 3 1

After this change is made the listing appears as follows (assuming that there are no changes which affect previous instructions):

6	PRCOMM	CLA	1, 4	GET PRINTER CONTROL WORD
7	+1	TPL	* +2	
8	+2	WPDA		DOUBLE SPACE PRINTER IF
9	+3	WPDA		CONTROL NEGATIVE, SINGLE IF +
10	+4	SLT	1	
11	+5	SXA	RESTOR, 1	SAVE INDEX
12	+6	SXA	RESTOR +1, 2	REGISTER

2.3.1.2 ALTER

The ALTER pseudo-operation is analogous to CHANGE in that it may occur in two forms similar to those of CHANGE and may be used to make insertions, deletions or both. ALTER, however, inserts and/or deletes the equivalents of symbolic source program cards instead of machine words.

There are two permissible forms for ALTER. The first is when N_1 and N_2 represent alter numbers.

1	Location 2 6	7	Operation 8 14	15	Address, Tag, Decrement/Count 16
			ALTER		N_1, N_2

This form indicates that the information corresponding to alter numbers N_1 through N_2 , inclusive, is to be deleted from the program. If symbolic cards are associated with an ALTER instruction in this form, the instruction also indicates that the cards are to be inserted into the program between $N_1 - 1$ and $N_2 + 1$. As with CHANGE, the number of insertions need not be equal to the number of deletions since the words following N_2 are automatically adjusted.

In the second form N is also an alter number:

	Location			Operation		Address, Tag, Decrement/Count
1	2	6	7	8	14	15 16
				ALTER		N

This form indicates that no deletions are to be made and that the associated program modification cards are to be inserted between the symbolic instructions numbered N and N + 1.

Restrictions:

- a) For an ALTER instruction in the first form, N_1 must be less than or equal to N_2 ; otherwise, the instruction and the symbolic cards to be inserted are ignored.
- b) Remark cards DETAIL, LIST, TITLE and UNLIST pseudo-instructions cannot be deleted by an ALTER. When an ALTER specifies alter numbers which include one of these in their range, the ALTER does not affect the remarks cards or listing pseudo-instructions.
- c) An ALTER instruction cannot delete an END card without also inserting an END card.
- d) An ALTER cannot insert an END card without also deleting an END card. If an ALTER includes an END and does not specify the deletion of an END, the END to be inserted is ignored.
- e) If a programmer macro-instruction is inserted by an ALTER, the definition must also be included with the list of modifications. This does not mean, however, that the definition must be included with the same ALTER that is to insert the macro-instruction. Instead, it may be included by a CHANGE or by another ALTER. The definition may also be placed in front of the group of modifications and need not be preceded by a Modify and Load pseudo-instruction.
- f) A modification by an ALTER must not overlap a modification either by another ALTER or by a CHANGE.

Example 1: Assume that the instruction is to be corrected with alter number 5 in the following listing:

```

4*
5x      ORG      START
6      PRCOMM    CLA      1, 4      GET PRINTER CONTROL WORD

```

The instructions necessary to accomplish the corrections are:

1	Location 2 6	7	Operation 8 14	15	Address, Tag, Decrement/Count 16
			ALTER ORG		5,5 3000

After this correction is incorporated and, assuming no changes affecting the preceding remarks cards, the listing appears:

```

4*
5          ORG          3000
6      PRCOMM      CLA      1, 4      GET PRINTER CONTROL WORD

```

Example 2: Assume that in the following listing the instructions with alter numbers 92 and 93 are to be deleted.

```

91      NUMBER      EQU          24
92      NUMBER      EQU          12
93      ZERO        EQU          0
94      TSTBIT      PZE                      STORAGE FOR TEST BIT

```

The required instruction is:

1	Location 2 6	7	Operation 8 14	15	Address, Tag, Decrement/Count 16
			ALTER		92, 93

After this change is made the listing appears (assuming no modifications affecting preceding instructions) as:

```

91      NUMBER      EQU          24
92      TSTBIT      PZE                      STORAGE FOR TEST BIT

```

2.3.1.3 SYMBOL

The SYMBOL instruction permits the assignment of a location symbol to a word without requiring the deletion and subsequent insertion of the word.

There is one form of a SYMBOL instruction:

1	Location 2 6	7	Operation 8 14	15	Address, Tag, Decrement/Count 16
	B		SYMBOL		A + n

B represents a symbol of from one to six characters which is to become associated with the word previously assigned the relative location expression $A + n$ (use relative numbers only).

If SYMBOL is used to associate a location symbol with a word which already has a location symbol, the new symbol does not replace the old; instead, the two are made synonymous by an EQU instruction. However, if the symbol in the location field of the SYMBOL instruction has been previously defined in the program, it is defined again with the new value and becomes a doubly-defined symbol.

If the location field or the variable field of a SYMBOL instruction is blank, the instruction is ignored.

When a SYMBOL instruction is to assign a symbol to a word in a headed area (for example, when A is headed) the instruction is written:

	Location			Operation			Address, Tag, Decrement/Count
1	2	6	7	8	14	15	16
	B			SYMBOL			H\$A + n

H is the character by which A is headed, and B and $A + n$ are as described previously.

Restrictions: If a principal pseudo-operation appears in the range H\$A and H\$A + n, inclusive (or if A is unheaded) the SYMBOL pseudo-instruction above has no effect upon the program.

Example: Assume that a symbol must, for convenience, be assigned the instruction with alter number 25 in the following listing:

16	CON6	PDX	6, 2	
17	+1	LGR	18	COMPUTE # INSERT WORDS -
18	+2	ADD	1, 4	START ADDRESS AND
19X	+3	STO	/////	STORE.
20	+4	CLA	CON6	INITIALIZE FOR OCTAL IF TAG
21	+5	TQP	*+2	OF PRINT CONTROL IS 4.
22	+6	ARS	1	IF OUTPUT IS OCTAL STORE
23	+7	STA	STRTWD-2	3 IN CONVERSION ADDRESS
24	+8	CLA	SWITCH	
25	+9	LLS	0	
26	+10	STO	SWITCH	
27	+11	AXT	24, 1	
28	+12	TCOA	*	DELAY UNTIL CHANNEL AVAILABLE
29X	+13	NOP	WKAREA+23,1	
30X	+14	STZ	WKAREA+24,1	CLEAR WORK AREA FOR
31	CLEAR	TIX	*+1, 1, 1	CONVERSION

1	Location 2 6	7	Operation 8 14	15	Address, Tag, Decrement/Count 16
	SHIFT		SYMBOL		CON 6+9

The symbol instruction above would appear in a subsequent listing (assuming no other changes) as:

16	CON6	PDX	6, 2	COMPUTE # INSERT WORDS -
17	+1	LGR	18	START ADDRESS AND
18	+2	ADD	1, 4	STORE.
19X	+3	STO	/////	INITIALIZE FOR OCTAL IF TAG
20	+4	CLA	CON6	OF PRINT CONTROL IS 4.
21	+5	TQP	*+2	IF OUTPUT IS OCTAL STORE
22	+6	ARS	1	3 IN CONVERSION ADDRESS
23	+7	STA	STRTWS-2	
24	+8	CLA	SWITCH	
25	SHIFT	LLS	0	
26	+1	STO	SWITCH	
27	+2	AXT	24, 1	
28	+3	TCOA	*	DELAY UNTIL CHANNEL AVAILABLE
29X	+4	NQP	WKAREA+23, 1	
30X	+5	STZ	WKAREA+24, 1	CLEAR WORK AREA
31	CLEAR	TIX	*+1, 1, 1	FOR CONVERSION

2.3.1.4 ASSIGN

The ASSIGN pseudo-instruction is provided so symbols may be defined or redefined by insertion of EQU, SYN or BOOL cards. The form of an ASSIGN instruction is illustrated below:

1	Location 2 6	7	Operation 8 14	15	Address, Tag, Decrement/Count 16
			ASSIGN		H

H represents a heading character which may be a blank.

This instruction must be followed by at least one EQU, SYN or BOOL instruction to perform one of the following functions:

- To define new symbols and undefined symbols in a program.
- To redefine symbols originally defined in a program by EQU, SYN or BOOL instructions.

An ASSIGN instruction may not be followed immediately by any instruction other than EQU, SYN, BOOL or SYMBOL. (Note that a SYMBOL following an ASSIGN does not terminate the effect of the ASSIGN.)

If an ASSIGN instruction specifies a non-blank heading character, all the symbols used in the following EQU, SYN and BOOL instructions are headed by that character. (In addition, the only EQU, SYN and BOOL instructions processed are those for which the location symbol has been previously defined by an EQU, SYN or BOOL card and is not multiply defined. Under these conditions, the new definition replaces the old one.)

When an ASSIGN specifies a blank heading character, the EQU, SYN and BOOL instructions are treated as follows:

- a) If the symbol in the location field of a SYN, EQU or BOOL instruction is undefined or is new to the program, the symbol becomes defined as usual. The EQU, SYN or BOOL instruction defining the symbol is inserted at the beginning of the program, preceded only by remarks included at the beginning of the source program deck.
- b) If the symbol in the location field of a SYN, EQU or BOOL instruction is defined in the new program by a SYN, EQU or BOOL and is not multiply defined, the new definition replaces the old one at the same point in the program.
- c) In all other cases the symbol in the location field of a SYN, EQU or BOOL instruction is multiply defined in the program.

When a SYMBOL card follows an ASSIGN, the location symbol is headed by the heading character of the ASSIGN, providing the location symbol is less than six characters long.

The symbol in the variable field of the SYMBOL is also considered headed under the same condition.

Example: Assume that the symbols WKAREA and IMAGE are to be equated in a program. The instructions necessary are:

Location	Operation	Address, Tag, Decrement/Count
WKAREA	ASSIGN SYN	IMAGE

The listing might then appear as follows (assuming no modifications affecting four remarks cards at the beginning of the source program):

4*			
5	WKAREA	EQU	IMAGE
6X		ORG	START

2.4 DEBUGGING MACROS

The principal function of debugging macros is to permit the programmer to investigate the contents of storage or control panel during the execution of his program. The debugging macros are used during the development phases of a program and are removed after debugging has been completed.

The debugging macros can be thought of as extensions of the pseudo-operations of the SCAT source language and, as such, can be inserted into the program either while coding or as modifications during Modify and Load. In the latter case, use of the ALTER pseudo-operation inserts the debugging macro at the desired location in the program.

When the Compiler or Modify and Load interprets a debugging macro, a TXL branch occurs to the debugging subroutine, the function called for is executed, all main program indicators and storage cells are restored, and program control returns to the main program at the instruction immediately following the debugging macro.

The programmer has the option, through the use of debugging macros, of specifying the format of information to be printed and whether or not he wants his data to be printed on-line or off-line.

It is evident that there are several categories of debugging macros. The types that call for the outputting of information are called information macros; those that specify the output format are called modal macros; the macros that permit selectivity of outputting are called conditional macros.

2.4.1 Variable Fields

Since the function and area of activity of debugging macros are varied, the variable field of the macros is tailored to the function of each. In general, the variable fields contain three types of information: location, format and count:

- a) Location--specifies the proper area of activity for core storage cells and the index.
- b) Format--indicates the format required for output results of information macros; specified format of VFD-introduced blocks of storage.
- c) Count specifies:
 - 1) Position of binary point in fixed point format.
 - 2) Number of times a conditional macro is satisfied or unsatisfied.
 - 3) Increment of every conditional macro.

A typical debugging macro may be:

<u>Location</u>	<u>Operation</u>	<u>Variable Field</u>
	CORE	L ₁ , L ₂ , F, IT ₁ , IT ₂

L₁ = First location (absolute or symbolic)

L₂ = Second location (absolute or symbolic)

F = This field designates the format of the output and may be coded as follows:

<u>Format</u>	<u>Code</u>
Symbolic Instruction	S
Fixed Point Number	X
Floating Point Number	F
Octal Integer	O
Hollerith BCD Information	H
Variable	V

IT₁ (or T₁I)--Indirect addressing and the tag information for the first location.

IT₂ (or T₂I)--Indirect addressing and the tag information for the second location.

2.4.2 Information Macros

CORE L₁, L₂, F, IT₁, IT₂: Execution of this macro causes the outputting of the core memory block from the lower location (defined by L₁ and IT₁) to the upper location (defined by L₂ and IT₂) in the specified format.

If the effective upper location is zero, the block of core memory from the lower effective location to the top of core memory is outputted. If the effective upper location is non-zero, it must not be less than the effective lower location.

PANEL (no variable field): Execution of this macro causes the outputting of:

- a) Accumulator and MQ--each in both octal and floating point format.

- b) Index Registers 1, 2 and 4--each in both octal and decimal.
- c) Sense Indicators--as an octal number whose binary equivalent has 0's for indicators which are off and 1's for those which are on.
- d) Sense Lights and Sense Switches--each a binary number, with 0's for those which are off or up, and 1's for those which are on or down, respectively.
- e) Entry Keys--as an octal number.
- f) Accumulator Overflow, Divide Check, and Input/Output Check Indicators--each either on or off.

2.4.3 Modal Macros

There are two modal macros with variable fields (FORMAT and POINT) and three without (ON, OFF and NUCASE). These macros set modes which, until countermanded by other modal macros, influence subsequently executed debugging macros.

FORMAT B₁, F₁, B₂, F₂..., B_n, F_n: Execution of this macro gives a desired meaning to the format code V.

If a block of words has been compiled by VFD pseudo-instructions, or otherwise involves a heterogeneous format, information macros output these words in proper format if the macros stipulate the format code V and if meaning has been given to this format code by prior execution of a suitable FORMAT macro. For example if locations B through B+2 contain the octal words 004003040010, 764240000000 and 254560000000, respectively, the macros:

FORMAT	6,O,15,X,25,X,12,H
CORE	B, B+2,V

cause the first (leftmost) six bits of location B to be outputted as an octal number, the next 15 bits as a fixed point number, the next 25 bits (continuing to location B+1) as a fixed point number and the next 12 bits as Hollerith information. For Hollerith (H), the number of bits should be a multiple of six.

POINT N: Execution of this macro defines the position of the binary point within a word to be outputted in fixed point format.

N is an integer from 0 to 35 which indicates the number of bits which lie to the left of the binary point.

ON (no variable field): After execution of this macro, and prior to execution of OFF or NUCASE, debugging information is printed on-line, written on the BCD output tape for peripheral printing.

OFF (no variable field): After execution of this macro, and prior to execution of ON, debugging information is written on the BCD output tape for off-line printing. This is the normal condition which prevails prior to execution of any ON macro and after NUCASE.

NUCASE (no variable field): If a program remains in core memory while it is repeatedly executed for different cases, occasioned, for example, by new data cards being read into a fixed area of core memory, the NUCASE macro executed at the start of each such case resets the POINT and ON modal macros to normal and resets all counts generated by count type conditional macros to zero, in addition to outputting a case identification number.

2.4.4 Conditional Macros

WHEN: When the variable field conditions are satisfied, subsequent information macros will be executed.

UNLESS: When the variable field conditions are not satisfied, subsequent information macros will be executed.

AND: This macro connects conditional and information macros and extends the power of the WHEN and UNLESS macros. For example, it may be difficult to specify both upper and lower limits of a given variable with one WHEN macro. However, if one limit is specified by a leading WHEN macro, the other limit can be specified by a following AND macro. When using the AND macro with a WHEN macro, both conditions must be satisfied.

OR: This macro connects a conditional and an information macro and extends the power of the WHEN and UNLESS macros. Unlike the AND macro, the OR macro permits the specification of more than one condition, any one of which permits the execution of subsequent information macros.

EVERY N: This macro specifies the increment of outputting for successive passes through a program loop. Its variable field consists of an integer N which allows a succeeding information macro to be executed the first time, and subsequently every Nth pass through a program loop.

Variable Fields of Conditional Macros: All conditional macros except EVERY can use the following general format for their variable fields:

$$L_1, R, L_2, IT_1, IT_2$$

The relation subfield R is coded in one of the following ways:

<u>R Code</u>	<u>Meaning</u>	<u>Comparison Employed by (DB)</u>
L	Less than	CAS
E	Equals	CAS
G	Greater than	CAS
LL	Logically less than	LAS
LE	Logically equals	LAS (redundant to E)
LG	Logically greater than	LAS

The other subfields involve some additional conventions peculiar to the conditional macros.

The rules governing the use of L_1 , L_2 , IT_1 and IT_2 are similar but not identical to those governing the CORE macro. The following rules govern the use of terms in the variable field (assume that OP is a WHEN, OR, UNLESS or AND macro):

a) Case 1; OP (L_1)

- 1) When L_1 is zero or blank the macro is meaningless.
- 2) When L_1 is 1 through 7 the contents of XR 1,2 or 4 or their result is specified.
- 3) When L_1 is greater than 7 a core storage cell is specified.

(NOTE: The above explanation is true if and only if at least one other subfield in the variable field is expressed. Otherwise, this form of the macro expresses a count type condition, for example, WHEN N, where N may be any number. See programming examples d, e and f.)

b) Case 2; OP L_1, R

Recall that R expresses a relationship between L_1 and the term that follows, e.g., L_1, E, L_2 (L_1 equals L_2).

- 1) All rules of Case 1 for L_1 apply.
- 2) R must be one of the six symbols established for the desired relationship.
- 3) Since L_2 is not expressed, the relationship specified is between L_1 and zero.

c) Case 3; OP L_1, R, L_2

- 1) All rules of Case 1 apply to both L_1 and L_2 .

2) R must be one of the accepted six symbols.

d) Case 4; OP L₁, R, L₂, IT₁

- 1) If I is written anywhere in the fourth term, L₁ is indirectly addressed.
- 2) L₁ is not inferred as an XR but is always a core storage location, regardless of magnitude.
- 3) If L₁ is blank it is regarded as a tagged address of zero.

e) Case 5; OP, L₁, R, L₂, IT₁, IT₂

All rules governing L₁ apply to both L₁ and L₂, using IT₂ as address modification specification for L₂.

2.4.5 Programming Examples of Debugging Macros

a) A STO X
CORE
B CLA Y

After execution of STO X, all of core storage is outputted on BCD tape in addition to the panel information and, immediately following, control is returned to CLA Y.

b) Given: L50 is the location of 50

WHEN 4, G, L50
CORE

If the contents of XR 4 are equal to or less than L50, the core macro is not outputted; only when the contents of XR 4 are greater than 50 is core memory outputted.

c) Given: Location 4 contains PZE 888; XR 2 equals 1

UNLESS 4, L, 2, I
OFF

If the contents of LOC 888 are equal to or greater than zero, the subsequent output is on-line. Conversely, if the effective address of location 4 is positive, subsequent output is on-line.

- d) WHEN 8 (count type condition)
CORE 800,800,X

If this pair is inserted in a program loop for the first seven executions of WHEN, CORE is inoperative; following the eighth execution of WHEN, CORE becomes operative.

- e) UNLESS 8 (count type condition)
POINT 18
CORE 800,800,X

If this sequence has been inserted in a program, the number of location 800 is a fixed-point integer, properly outputted until eight outputs have occurred. Thereafter the sequence is inoperative, however.

- f) WHEN 3 (count type condition)
UNLESS 3 (count type condition)
CORE A,A

If this sequence has been inserted in a program loop, the CORE macro becomes inoperative on the first and second passes, outputting occurs on the third, fourth and fifth passes, and all subsequent passes are inoperative.

- g) Given: X is to be outputted whenever it lies between 50 and 70. 50,X and 70 are located at L50, LX and L70, respectively.

The macro program to give proper output can be written:

```
WHEN L50,L,LX
AND LX,L,L70
CORE LX,LX,X
```

- h) Using the given locations in Example "g," it is required to output X when it is less than 50 or greater than 70; the following macro accomplishes this:

```
UNLESS L50,L,LX
AND LX,L,L70
CORE LX,LX,X
```

- i) Given: X is to be outputted when $X^2 \geq 100$. X,10 and -10 are located at LX,L10 and LM10, respectively. The coding is:

```
WHEN LX,L,LM10
OR LX,G,L10
CORE LX,LX,X
```

- j) Given: The first 50 non-negative values of X in a loop are to be outputted. The coding is:

```
UNLESS LX,L,0
OR      50          (count type condition)
CORE   LX,LX,X
```

Here the UNLESS macro is associated with a count of outputs.

- k) Given: X,Y and Z are located at LX,LY and LZ, respectively. X is to be outputted the first 50 times that any of the following three conditions are satisfied:

```
X exceeds Y and XR 4
X exceeds XR 2
X exceeds Z and XR 4
```

The coding is:

```
WHEN   LX,G,LY
OR     LX,G,LZ
AND    LX,G, 4
OR     LX,G, 2
UNLESS 50
CORE   LX,LX,X
```

(For a detailed explanation of the associative and commutative laws governing this type of sequence see page 29, Part 3, of the SHARE 709 SOS Manual.)

2.5 MONITOR

The Monitor is a supervisory program written to control the processing of job decks through the computer. A job deck consists of a program deck and its associated control cards which designate the operation to be performed.

The control cards direct the Monitor to perform any or all of the following:

- a) Compile a program (listing and squoze deck as output).
- b) Modify and load a squoze deck for execution.
- c) Modify and punch a squoze deck to punch a clean (no modifications) squoze deck.
- d) Produce a listing of a squoze deck with or without modifications.
- e) Permit the use of debugging macros.

2.5.1 System Operation: Input Deck

When using the SOS system for an assembly, a debugging run or an execution run, the first card of each job deck is a JOB card. The alphabetic characters J, O and B are punched in columns 8-10 of the card. Also punched in columns 16-27 of the card are the name of the program and the programmer's name or initials -- to enable the operator to separate and return the results.

The input deck consists of any sequence of job decks, followed by a card punched PAUSE in columns 8-12. Job decks include the following possible categories:

a) Compilation Job Decks

- 1) Card punched JOB in columns 8-10 with the name of the program and the programmer (or his initials) in columns 16-27. Columns 11-15 must be blank.
- 2) Card punched CPL in columns 8-10 for column binary, or CPLRB in columns 8-12 for row binary output.
- 3) At least two remark cards, one with the name of the program and one with the name of the programmer.
- 4) Symbolic program deck from ORG to END card.
- 5) Blank card
- 6) PAUSE card

Non-modified, column or row binary squeeze decks may be inserted in the symbolic deck if preceded immediately by a SQZ symbolic card. For column binary, SQZ is punched in columns 8-10; for row binary, SQZbRB is punched in columns 8-13. The squeeze decks incorporated in the symbolic deck must be complete.

b) LS: List Job Deck

- 1) JOB card (as in a above)
- 2) Cards punched LS in columns 8 and 9
- 3) Squeeze deck without modification
- 4) Blank card
- 5) PAUSE card

c) Execution Job Deck

- 1) JOB card
- 2) Card punched LG in columns 8 and 9
- 3) Squeeze deck*
- 4) Blank card
- 5) Any number of data sentence decks**
- 6) Card punched GO in columns 8 and 9
- 7) PAUSE card

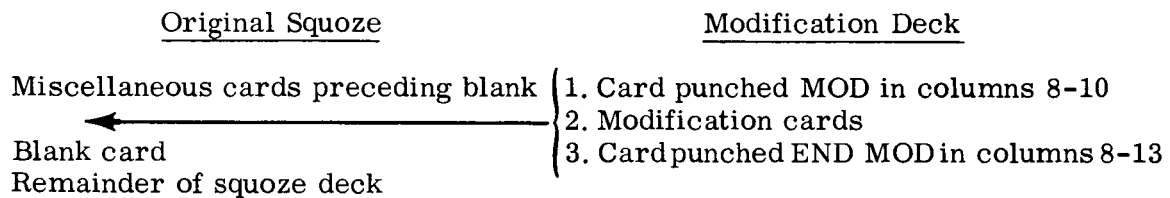
The card sequence with a squeeze deck is of major importance; manual rearranging should be avoided. When no modification is desired there is no change required in the squeeze deck; it should be fed into the card reader exactly as produced in the card punch.

d) List Squeeze Deck

This job deck gives a dump-type listing of the squeeze deck with modifications. The listing does not contain any comments, but it looks like a dump using the CORE macro, with the symbolic format specified.

- 1) JOB card
- 2) Card punched LG in columns 8 and 9

* If modifications are to be added, they are to be inserted as shown below:



** Data sentence decks are composed as follows:

1. Card punched DS1 in columns 8-10
2. Data sentence decks
3. Blank card

Data sentence decks may be used to provide for input data during the debugging of programs. Additional information concerning DS1 cards is found in Subsection 2.5.3.

- 3) Squeeze deck with modifications
- 4) Blank card
- 5) Card punched LIST in columns 8-11
- 6) PAUSE card

This type of deck must be used if there are no modifications.

e) PS: Punch New Squeeze Deck

- 1) JOB card
- 2) Card punched PS in columns 8 and 9
- 3) Squeeze deck with modifications
- 4) Blank card
- 5) PAUSE card

f) PA: Punch Absolute Binary

The following job deck causes the squeeze deck to be decoded and absolute binary cards to be punched according to SOS format.

- 1) JOB card
- 2) Cards punched PA in columns 8 and 9
- 3) Squeeze deck with or without modifications
- 4) Blank card
- 5) PAUSE card

g) Compile and Execute

h) Punch New Squeeze Deck and Execute

i) List Squeeze and Execute Deck

2.5.2 Effect of Control Card

<u>Control Card</u>	<u>System Action Caused</u>	<u>Visible Results</u>
JOB	Initializes the Monitor and causes Monitor to read next card.	Prints JOB and remarks from JOB card variable field on-line.
CPLRB	Calls in the Compiler and transfers control to the Compiler. The Compiler compiles the program, gives an error list and punches a squoze deck. Control is then transferred to the Monitor, which reads in Modify and Load to obtain a Modify and Load error list and a program listing.	Prints CPLRB on-line and off-line. Pass SYSTAP to C1 and C2. When C2 is in, the system tape re-winds. Prints error list on-line or off-line; punches squoze on-line or off-line in row binary.
CPL	Same as for CPLRB	Same as for CPLRB, except the squoze deck is punched in column binary.
PS	Calls in Modify and Load, punches a new squoze deck and gives a program listing. May be used with or without modifications. MOD and END MOD cards must be used even if no modifications are present.	Prints PS on-line and off-line. Punches new squoze on-line or off-line. Gives new program listing on-line or off-line.
LS	Calls in Modify and Load and gives a listing. No modifications are permitted.	Prints LS on-line and off-line. Gives program listing on-line or off-line.
LG	Calls in Modify and Load, transfers control to Modify and Load, decodes squoze and writes absolute program on B1. At end of loading, it transfers control to Monitor to read next control card. Modifications are permitted.	Prints LG on-line and off-line.

<u>Control Card</u>	<u>System Action Caused</u>	<u>Visible Results</u>
PA	Calls in Modify and Load, decodes squeeze and writes absolute program on B1; then punches absolute binary. Mods are permitted.	Prints PA on-line and off-line. Punches absolute.
GO	Reads SNAP (the DB1 program) into core memory below 5670 ₈ ; clears memory from 5670 ₈ to 0; loads program from B1 until a transfer card record is read; then transfers control to object card program.	Prints GO on-line and off-line.
LIST	Reads SNAP (DB1) into core memory below 5670 ₈ ; clears memory from 5670 ₈ to 0; loads program from B1 until a transfer card record is read; executes core dump from 5670 ₈ to 0; then transfers control to Monitor to read next control card.	Prints LIST on-line and off-line.
PAUSE	Halts Monitor and allows continuing without rewinding all tapes. Press START to read next control card.	Prints PAUSE on-line and off-line.
STOP	Rewinds all system tapes (B1, B2, A1, A2, A3, A5) and halts machine. Cannot re-start.	Prints STOP on-line and off-line.

2.5.3 Specifications of the Data Sentence Program

A data sentence is defined to include an absolute decimal location giving the initial loading address; this is terminated by an equals sign (=) which is followed by the data. Consecutive words of data are separated by commas until the end of the sentence, indicated by the marker (*); for example, 7083 = -52,32. 1E5,39.1B6* is a data sentence which loads three numbers—integer, floating and fixed numbers—into location 7083 and the two locations following.

The normal sentence data is floating point data, fixed point data and decimal integers which are expressed according to regular SCAT rules and which may follow each other arbitrarily.

To introduce octal data, the letter O is punched with the octal numbers enclosed within parentheses; for example, 7083 = -52,32. 1E5,39. O(-7,7263), 509E20*. This sentence loads three decimals, two octals and one decimal beginning at 7083.

The remaining rules of syntax are:

- a) The card is used from column 1 to column 72; punching is continuous.
- b) A sentence may start in any card column and extend to the end-of-sentence marker. It may extend beyond a card; more than one sentence may appear on a card.
- c) Punching on a card must end with a comma or with an end-of-sentence marker. If a blank then follows, the remainder of the card is ignored.
- d) The last sentence of a data block must end with a (\$) instead of (*) and should be followed by a symbolic expression. Transfer to this location is made after loading the data block; for example:

Card 1-A = 7192 = 5.1E3,60.12,301.2*
Card 2-B = 7195 = 70.1,O(-77),70,1B7\$C

These two cards comprise a data block, which load as specified and transfer to location C.

Two types of errors may occur during conversion:

- a) Overflow/Underflow--normal zero is stored; conversion of next field continues.
- b) Mispunch--when an illegal character is encountered, normal zero is stored and processing is continued for the next field.

Error messages indicating column number and record are given.

If either TCD's or DS1's are used the program must anticipate the logical record arrangement and call program and data blocks after logical record 1 from tape into core memory by use of calling sequences of the form:

TSX 82, 4
PZE A, , B
Bad data return

A is the number of the desired logical record. A = 0 means to read the logical record with the number that is one greater than the last one read. A non-zero B is the location to which the Monitor returns after reading. B = 0 causes return to the location specified by the TCD, END or \$ card.

Section 3

OTHER PROGRAMMING STANDARDS

3.1 MERCURY PROGRAM WRITEUP SPECIFICATIONS

A standard for program writeups was established at the beginning of Project Mercury to prevent duplication of effort and to systematize the work of the programming group. Clarity and readability are the goals of program documentation, and effective organization is the means to achieve them. However, there is no ironclad rule or outline for building a system; the arbitrary selection of an outline for writing programs is a problem in semantics. The general outline of program writeups (below) is an example of this problem, since the writeup as an entity should indicate how the program is to be used, or the method of usage. All program writeups (of Monitor, Processor, External and Simulation routines) conform to the following breakdown:

TITLE
Introduction
Input Requirements
Output Requirements
Method
Usage

Within the framework presented above are many subcategories, each of which is not necessarily applicable to every routine. The detailed breakdown outlined below attempts to combine all possible subgroups within the general outline. Textual material explains the application of a major heading and, by implication, its subcategories. The specific type (or types) of routines to which a subclass may apply is also indicated. However, only the general outline applies to all programs; there is no similar rule for subcategories—deviations are commonplace.

(NOTE: In the following discussion the term “program” refers to that set of machine instructions and pseudo-operations (BSS, BES, DEC, OCT, etc.) which comprise a “running” deck, i.e., a deck that is ready to be “run” on the computer, including the necessary control cards with the symbolic deck.)

X.X TITLE

Introduction—including the purpose and performance of the program and its place in the system.

X.X.1 Input Requirements

All information and/or conditions accepted by a given program and inherent to the execution of its stated purpose are input requirements. Examples are:

RequirementsUsed By*

Programs	M P S
Subroutines	M P E S
Library Subroutines	M P E S
System Macros Defined	M
Programmer Macros Defined	M
Constants (KXXXXX)	M P
Tables (TMXXXX)	M P
Communication Cells (MCXXXX)	M P
Program Parameters**	M P
Symbolic Locations for Storage***	M
Inputs from Radars through DCC	P
Conditions on Entry (AC,MQ,XR's, Indicators)	M P E

X.X.2 Output Requirements

All information and/or conditions transferred out of a given program and inherent to the execution of its stated purpose are output requirements. Examples are:

RequirementsUsed By

Tables (TXXXXX)	M P
Communication Cells (MCXXXX)	M
Program Parameters	M P
Conditions on Exit (AC,MQ,XR's,Indicators)	M P E

X.X.3 Method

Those elements of mathematical methodology (equations, formulas, etc.) which are necessary contributors to the performance of the program are included in "Method." This can be applicable to Monitor, Processor and External routines.

X.X.4 Usage

This section refers specifically to the mechanical operation of the program and indicates briefly the statistics necessary to the purposeful execution of the program. Examples are:

* The following codes apply to all listings: M - Monitor, P - Processors, E - External routines and S - Simulation.

** Numerical information used only with the given program.

*** Data locations used by as many programs as necessary and defined somewhere in the system. Called "common" by External routines.

<u>Data</u>	<u>Used By</u>
Entry From	M P
Exit To	M P
Calling Sequence	M P E
Storage Required	M P E S
Instructions	M P E S
Macros	M
Temporary	M P E
Subroutines	M
Cells	M
Tables (TMXXXX)	P
Common (MCXXXX, TCXXXX—Monitor)	M P E
Timing	M P E S
Error Codes	P E
Checkout	P E S
Operator's Notes	E S
Accuracy	P E
Exempt Symbol (exempt from relativization)	E

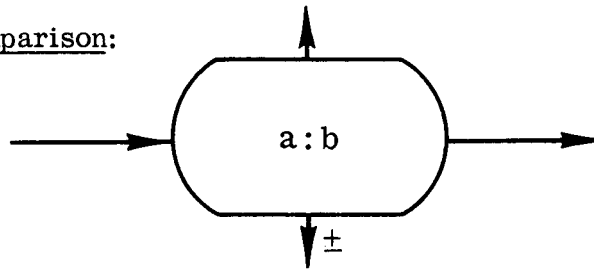
3.2 FLOW CHARTING STANDARDS

Flow charting standards were suggested to facilitate communication between programmers and to maintain clarity and consistency. In many cases, however, programmers used individual flow charting techniques. The symbols pictured in this subsection are those recommended in the SHARE Reference Manual and reflect ideas generally advanced by Von Neumann and Goldstine. All the symbols presented here are provided on the IBM Programming Template, Form X24-5884-5.

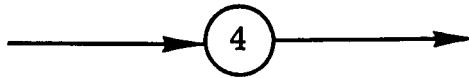
Operation, Function:



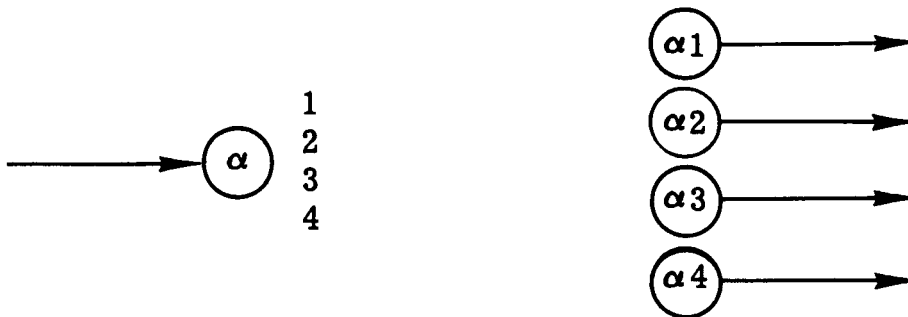
This symbol indicates the uninterrupted storing of instructions which involve some form of data transformation—formulas, substitution expressions, input or output. For input and output functions this box should indicate the process initiated, the unit used, format, etc. The completion of the process should be tested in a decision box.

Decision, Comparison:

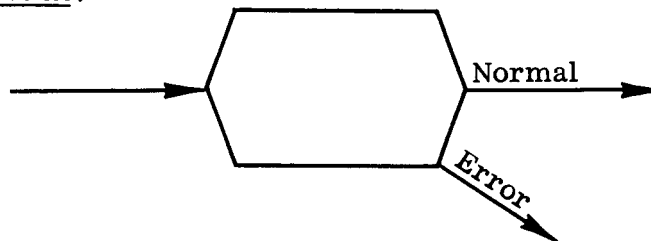
This symbol is used for conditional, or branch, operations. For example, $a:b$ means compare a with b relative to the relationships specified on the exit lines. This box can handle any test of yes/no, on/off, bit/no bit, etc. Of course, b can be any variable or fixed quantity.

Fixed Connector:

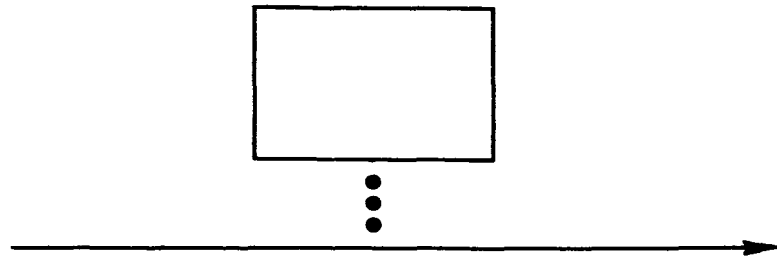
This symbol is used to connect parts of a flow chart to avoid crossing lines. The designation in the circle should refer to box labels. This type of connection should be used as a merge point when more than one entry into an operation box is needed.

Variable Connector:

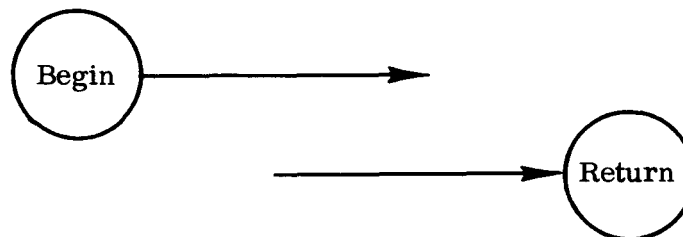
This symbol is used to indicate a switch having several alternate paths. The switch is set previously in an operation box by $a_1 \rightarrow a$ or $a = a_1$. The arrow (\rightarrow) means "replace"; the equals sign ($=$) means "is set equal to".

Closed Subroutine:

This symbol indicates the use of a subroutine. The box should be referenced pictorially to either a subroutine writeup or a flow chart.

Assertion:

This symbol denotes explanatory information concerning coding tricks, definitions of new names, data changes or requirements, or describes the logic function which follows.

Entries and Exits:

Since every program can be considered a subroutine to some other program, BEGIN and RETURN should be used for all programs. The calling sequences should appear on the flow chart. Multiple RETURN's should be handled by using a RETURN as a variable connector.

3.3 PROGRAM CHECKS

Program checks are necessary for maintaining consistency in combining one program with another so they may merge or separate easily.

Before a completed program was given to the librarian to be filed, the following items were checked by the Standards Group:

- a) A deck containing the six-character symbols used by the program, i.e., those shown in the symbol listings of the constants used.
- b) The job history of the program, including tests made, etc.
- c) A preliminary writeup and listing.

After the program was checked, the following information was given to the librarian:

- a) Correct symbolic deck
- b) Squeeze deck
- c) Three copies of the listing (LS)
- d) Completed writeup to be checked and typed
- e) Flow chart
- f) Sample output, if available.

3.4 SYMBOLS

Project Mercury subroutines, tables and data locations are each denoted by six-character symbols. The first two characters are determined by the method described below; the last four characters are used to assign mnemonic values to a symbol. Any symbols used exclusively within a given subprogram and having no meaning within the system as a whole are limited to five characters or less. At his discretion the programmer may or may not use either or both of the two identifying characters listed below.

All subroutines, tables and data locations (including constants, parameters and communication cells) are given system symbols. The first identifying character is always alphabetic and is mnemonic wherever possible; the second character is numeric for all sections except Monitor, Simulator, Bermuda and Cape Canaveral. For these latter four sections, the second character may be alphabetic and may refer to the first character of the other sections. The numeric characters are assigned in sequence as the subroutines are developed.

The following are examples of prefix assignments:

A0	(Main)	Acquisition Data Generator
A3		Launch/Abort Computation of Latitude and Longitude
B	(Main)	Bermuda
BA		Bermuda Short Arc Orbit Determination
BE		Bermuda Editing
BH		Bermuda Herget's Method Computations
BI		Bermuda Input
BO		Bermuda Output

BR		Bermuda Retrofire Calculations
BS		Bermuda Smoothing
CC	(Main)	Cape Canaveral Launch/Abort Processors
C9		Canaveral Retrofire Calculations
CH		Canaveral Herget
D0	(Main)	Differential Correction
D1		Set Up Equation of Least Squares
D2		Modified Least Squares
D3		D.C. Interval Determination
D4		Calculate and Convert R and V to Orbit Parameters
D5		Residual Block Calculations
E0	(Main)	Edit
F0	(Main)	Time to Fire Retrorockets
H0	(Main)	Herget's New Method
I0	(Main)	Input
I1		Transmission File
I2		Log on Tape
K		Constants
M0	(Main)	Monitor
MF		Monitor Suffix
MN		Name of Ordinary Processor
MP		Monitor Prefix
MS		Subroutine for Monitor Use
MT		Trap Processors

	MU	Input/Output Unit
	MY	Ordinary Processor Written by Monitor
MC		Communication Cell
MX		Routine External to the System
N0	(Main)	Numerical Integration
	N1	Set Up Function Table
	N2	Extrapolation and Correction
	N3	Calculate Second Derivations
	N4	Drag Acceleration
	N7	Variable Step Integration Equations
O0	(Main)	Output (displays and other output media)
P0	(Main)	Sliding Wire Impact Predictor
R0	(Main)	Re-entry
	R1	Numerical Differential Correction
	R2	Development of Numerical Integration Output Tables
	R5	Re-entry and Retrofire Calculations
	R6	Produces R, V and T at End of Retrofiring
S0	(Main)	Simulation
TM		Tables
U		Utility Programs (The second character is numerical and refers to the general function of the utility program.)
	U0	Conversions—Coordinate Systems, Units
	U1	Elementary Functions
	U2	Binary-to-BCD for On-line Output

U3	Vector Manipulations
W1POST (Main)	Postflight Analysis
W2POST	Launch
W3POST	Abort
W4POST	Orbit
W5POST	Re-entry

Section 4

MATHEMATICAL STANDARDS

4.1 TERMINOLOGY

The following alphabetical listing includes selected general astronomical terms, mathematical symbology and specific Project Mercury reference quantities. Some of the information, by implication, is common knowledge; other data was generated primarily for Mercury. All of the values and terms are used either supplementally to or directly with Project Mercury programming.

Symbols for Dimensions

L = Length

M = Mass

T = Time

K = Temperature

TERM	DIMENSIONS	SYMBOL	DEFINITION
Acceleration of Gravity	$[LT^{-2}]$	$g(H_0)$	The ratio of the weight of a material particle to its mass.
Apogee			Point on orbit farthest from the geocenter.
Argument of Perigee	$[Angle]$	ω	The angle between the ascending node and perigee on the celestial sphere.
Ascending Node Unit Vector	$[L]$	\bar{N}	Unit vector directed from geocenter toward ascending node.
Azimuth	$[Angle]$	A	The bearings of the capsule in the horizon plane of the station measured clockwise from north, $0 \leq A < 360^\circ$.
Coefficient of Drag		C_D	A number which relates to the retarding force experienced by a body in motion through fluid.
Date	$[T]$	$T_{(ref)}$	Equals zero. Reference time is midnight prior to launch.

TERM	DIMENSIONS	SYMBOL	DEFINITION
Density	$[ML^{-3}]$	$\rho(H)$	The ratio of the mass of a homogenous portion of matter to its volume.
Eccentric Anomaly	$[Angle]$	E_a	The angle included between perigee and the perpendicular projection of the capsule's instantaneous position from the major axis on the major auxiliary circle.
Eccentricity		e	Ratio of the center-to-focus distance to the semimajor axis, a .
Elevation	$[Angle]$	E	The capsule's angular distance above the horizon plane, measured from the station.
Ellipsoid Flattening		f_e	Geophysical constant related to the shape of the geoid.
Ephemeris			A time history tabulation of the position of the orbiting capsule with respect to its reference coordinate system.
Eta		η	Orbit element = $e \sin \omega$.
First Sum Vector	$[LT^{-1}]$	\vec{F}	First sum vector: (\vec{F}_x , \vec{F}_y , \vec{F}_z)
Geocenter			The point of intersection of the polar axis with the equatorial plane.
Geocentric Latitude	$[Angle]$	ϕ'	The angle included between the equatorial plane and a line joining the geocenter and a point on the surface of the earth, measured north or south, $-90^\circ \leq \phi' \leq +90^\circ$.

TERM	DIMENSIONS	SYMBOL	DEFINITION
Geodetic Latitude	[Angle]	ϕ	The angle defined by the intersection of a normal to the earth's surface with the equatorial plane, measured south or north, $-90^{\circ} \leq \phi \leq 90^{\circ}$.
Geometric Altitude	[L]	H	The increase in potential energy of a unit mass lifted from sealevel to a given altitude against the force of gravity.
Geopotential Altitude	[L ² T ⁻²]	H _g	
Geopotential at Base of Layer	[L ² T ⁻²]	H _b	
Inclination	[Angle]	i	Dihedral angle between the plane of the equator and the plane of the orbit.
Inertial Coordinate System		ICS	Coordinate system with origin at the geocenter (see X, Y, Z).
Inertial Longitude of Greenwich	[Angle]	λ_0	Hour angle of Greenwich at a reference time, t_0 .
Lateral Area	[L ²]		Frontal surface area of body; used in drag considerations.
Line of Nodes			The line determined by the intersection of the plane of the orbit of the capsule with the earth's equatorial plane.
Local Coordinate System		LCS	Local coordinate system at each radar station.
Longitude	[Angle]	λ	The arc of the equator included between the prime meridian and the meridian of the place, measured eastward $0 \leq \lambda < 360^{\circ}$.

TERM	DIMENSIONS	SYMBOL	DEFINITION
Longitude of Capsule	[Angle]	μ	Angle between the ascending node and the instantaneous position of the object, $\mu = N + \omega$ (mean anomaly of the node).
Longitude of Node	[Angle]	Ω	The arc of the celestial equator included between the vernal equinox and the ascending node, measured eastward.
Longitude of Perigee	[Angle]	π	Sum of longitude of node plus the argument of perigee. $\pi = \Omega + \omega$
Mean Anomaly	[Angle]	M_a	Angle between perigee and mean position.
Mean Longitude of Capsule	[Angle]	U	$U = \omega + M_a$. Angle between ascending node and mean position.
Mean Motion	[T ⁻¹]	\dot{n}	Average rate at which the orbiting capsule describes an arc.
Molecular-Scale Temperature	[K]	T_M	Mathematical variable introduced for theoretical reasons.
Normal-to-Orbit Plane	[L]	\bar{R}	Unit vector normal to orbit plane; sense determined by orbital angular momentum.
Orbit (parameter or element)			One of a set of quantities which completely describes an orbit.
Perigee			Point on orbit closest to the geocenter.
Perigee Unit Vector	[L]	\bar{P}	Unit vector directed from geocenter toward perigee.

TERM	DIMENSIONS	SYMBOL	DEFINITION
Perturbations			Deviations from two-body motion caused by atmospheric drag—the earth's equatorial bulge, etc.
Proportionality Constant (units depend on those of H)		G	
Radius of Earth at Equator	$[L]$	r	
Radius Vector	$[L]$	\bar{r}	Radius vector from geocenter to capsule in ICS.
Scale—height		H_S	Mathematical variable, negative reciprocal of the altitude derivative of logarithmic pressure.
Sealevel Value of g at Latitude ϕ (scalar)	$[LT^{-2}]$	g_ϕ	
Sealevel Value of g at Equator (scalar)	$[LT^{-2}]$	g_e	
Second Derivative Vector	$[LT^{-2}]$	$\ddot{\bar{F}}$	Second derivative vector: (\ddot{X} , \ddot{Y} , \ddot{Z})
Second Sum Vector	$[L]$	${}''\bar{F}$	Second sum vector: (${}''F_x$, ${}''F_y$, ${}''F_z$)
Semimajor Axis	$[L]$	a	One-half of the maximum chord of an ellipse.
Sidereal Period	$[T]$	P	Time an orbiting body requires for one complete revolution.
Slant Range	$[L]$	R	Distance from station to capsule in LCS. $R \geq 0$.
Slant Range Unit Vector	$[L]$	$\bar{\rho}^*$	Unit vector from station to capsule
Slant Range Vector	$[L]$	$\bar{\rho}$	Vector from station to capsule

TERM	DIMENSIONS	SYMBOL	DEFINITION
Speed	$[LT^{-1}]$	V	The ratio of distance to unit time.
Station Distance from Geocenter	$[L]$	R_s	
Temperature	$[K]$	T_K	Negative of the "lapse rate" —slope of the altitude—temperature profile.
Temperature Gradient (scalar)	$[KL^{-2}T^2]$	L_M	
True Anomaly	$[Angle]$	v	The angle measured from the center of the orbit in the direction of motion between perigee and the capsule position.
Unit Vector along X Axis	$[L]$	\bar{I}	
Unit Vector along Y Axis	$[L]$	\bar{J}	
Unit Vector along Z Axis	$[L]$	\bar{K}	
Value of T_M at Altitude H_b	$[K]$	$(T_M)_b$	Velocity vector in ICS.
Velocity Vector	$[LT^{-1}]$	\bar{V}	
Ξ		ξ	Orbit element = $e \cos \omega$
\bar{V} Component in X Direction	$[LT^{-1}]$	V_x	
\bar{V} Component in Y Direction	$[LT^{-1}]$	V_y	
\bar{V} Component in Z Direction	$[LT^{-1}]$	V_z	
X (ICS)	$[L]$	X	ICS axis directed from geocenter toward vernal equinox.
Y (ICS)	$[L]$	Y	
Z (ICS)	$[L]$	Z	ICS axis in earth's equatorial plane forming a righthand set with Z and X axes.
			ICS axis directed from geocenter toward north celestial pole.

4.2 COORDINATE SYSTEMS AND CONVERSIONS

There are six coordinate systems used by the NASA-Langley Space Task Group; two of these systems are the Local and Inertial Coordinate Systems (LCS and ICS) used for Project Mercury. The six systems, and the coordinates which each employs, are:

- | | |
|---|--|
| a) Burroughs-General Electric quasi-inertial | ξ, η, ζ |
| b) Spherical inertial | $\lambda_i, r, L_c, V_i, \gamma_i, \psi_i$ |
| c) Pod rectangular | u, v, w |
| d) IP 709 quasi-inertial | X, Y, Z |
| e) General Electric radar | $R, A, E(\text{LCS})$ |
| f) True Cartesian inertial, referenced to the first point of Aires. (\mathcal{T}) | $\bar{X}, \bar{Y}, \bar{Z} (\text{ICS})$ |

The coordinates for the Inertial Coordinate System in IBM notation are X, Y, Z.

The earth is neither a point source nor a homogeneous body and, therefore, does not reduce to a point attraction. Also, it is a rotating body and is non-spherical; hence, a simple potential function whose various first derivatives yield the components of the force cannot be deduced. This condition causes an expansion in terms of a trigonometric series whose coefficients are Legendre polynomials. The resultant force vector is thereby expressed. The leading coefficient is $\mu = 3.9860266 \times 10^{14} \text{ m}^3/\text{sec}^2$. There is no first harmonic; the second harmonic, J_2 , is $-1.755 \times 10^{25} \text{ m}^5/\text{sec}^2$; H , the third harmonic, is not used; $J_4 = -1.59 \times 10^{-6} \text{ m}^7/\text{sec}^2$ is the fourth harmonic.

In general, if ϕ is the potential of the earth's gravitational field at a distance r from its center and at a declination δ , then:

$$\phi = \frac{K_e^2}{r} \left[1 + \frac{1}{3} J_2 \left(\frac{a_e}{r} \right)^2 (1 - 3 \sin^2 \delta) + \frac{1}{5} H \left(\frac{a_e}{r} \right)^3 (3 \sin \delta - 5 \sin^3 \delta) + \frac{1}{30} J_4 \left(\frac{a_e}{r} \right)^4 (3 - 30 \sin^2 \delta + 35 \sin^4 \delta) + \dots \right]$$

where:

$$K_e = MG$$

G = universal constant of gravitation

M = mass of the earth

a_e = earth's equatorial radius

Mercury uses the International Ellipsoid which has an a_e value of 6.378145 x 10⁶ meters. The other values of the International Ellipsoid are:

a) Flattening: $f = 1/298.3 = 0.00335233$

b) Rotational speed of the earth: $\omega = .729211508 \times 10^{-4}$ rad/sec

c) Equatorial gravity: $g_e = 9.78034$ m/sec²

Using $a = 637814500$ cm., $f = 1/298.3$, and $g_e = 978.034$ cm/sec², then

$$g = \left(1 + J_2 + \frac{J_4}{2}\right) \frac{\mu}{a^2} - a\omega^2$$

when $\omega = 0.0000729211508$ rad/sec.

$$J_2 = f\left(1 - \frac{f}{2}\right) - \left(1 - \frac{9f}{7}\right) \frac{P}{2}$$

$$\text{and } J_4 = 3f\left(f - \frac{5P}{7}\right)$$

(In both of the previous cases, $P = \frac{a^3 \omega^2}{\mu}$) Therefore:

$$g_e = 978.034 = (1 + f + f^2) \frac{\mu}{a^2} - (a\omega^2)\left(\frac{3}{2} + \frac{3f}{7}\right).$$

Dividing 978.034 by a , each length becomes units of $a = 1$. In defining units of time, $T_{(\text{sec})}$ is such that $\mu = 1$, $a = 1$, and $f = 1/298.3$. However, when using the above J_2 and J_4 values, $T = 806.8104$ sec, using $a^3 = \mu T^2$ and solving the following equation:

$$\mu(1 + f + f^2) = 0.0000015334145T^2 + 0.00000000798388T^2.$$

(NOTE: An arithmetic error which yields a new T was noted. However, no change will be made at least until the ultimate spheroid, the DOD spheroid of 1960, is declassified.)

The mass of the earth is the unit of mass in H_g units and is 5.9765×10^{24} kilograms.

To correct observed data to a geocentric coordinate frame and, conversely, to compute acquisition data for each site, the position and coordinate system of each site must be accurately known. The reference system used in specifying geodetic latitude, longitude and altitude is, as specified by Cape Canaveral down-range findings and the U.S. Coast and Geodetic Survey, the Clarke spheroid of 1866.*

Page 496 of the American Ephemeris and Nautical Almanac, 1960, refers to the International Ellipsoid of Reference.** The American Ephemeris also provides the following for converting from geodetic latitude, ϕ , to geocentric latitude, ϕ' :

$\phi' = \phi - 11' 35'' 6355 \sin 2\phi + 1' 1731 \sin 4\phi - 0.0026'' \sin 6\phi$. The notation $11' 35'' 6355$ means 11 minutes, 35.6355 seconds of arc.

Local radius at a given (geodetic) latitude is given by:

$$\rho = a(0.998320047 + 0.001683494 \cos 2\phi - 0.000003549 \cos 4\phi + 0.000000008 \cos 6\phi)$$

The latter results if there is an expansion and ζ/a is accepted; then it is r-h in H_g units. Also from the American Ephemeris comes the value for the mean solar day: $1.0027379093 \times$ the mean sidereal day (p.495).

Since the Mercury orbit is a conic section, five quantities define its path; its position at a given instant determines its later position. The equations of motion (three in number) are of second order, each requiring two constants of integration. Thus, six constants are used to specify the motion completely.

When determining the relationship between time and place in orbit, the following elements are used:

- a) True anomaly: v
- b) Mean daily motion: $\eta \frac{2\pi}{P}$, when P = sidereal period.
- c) Eccentric anomaly E_a : $nt = E_a - e \sin E_a$
- d) Mean anomaly: $M_a = nt$. Therefore, $M_a = E_a - e \sin E_a$ (Kepler's equation).

In computing an ephemeris for the capsule's nearly-circular orbit, Herget introduced certain elements to overcome underflows of $e \approx 0$. These "Herget elements" are $\xi = e \cos \omega$ and $\eta = e \sin \omega$.

* $a_c = 6378206.4$ meters; $1/f_c = 294.979$; and $b_c = 6356583.8$ meters.

** $a = 6378388$ meters; $\frac{1}{f} = 297$; and $e^2 = 0.006722670022333322$.

The classical orbital elements are:

i = Inclination

Ω = Longitude of ascending node (Both i and Ω determine the orbital plane through the center of the earth)

a = Semimajor axis (or n = mean daily motion; $n = ka^{-3/2}$)

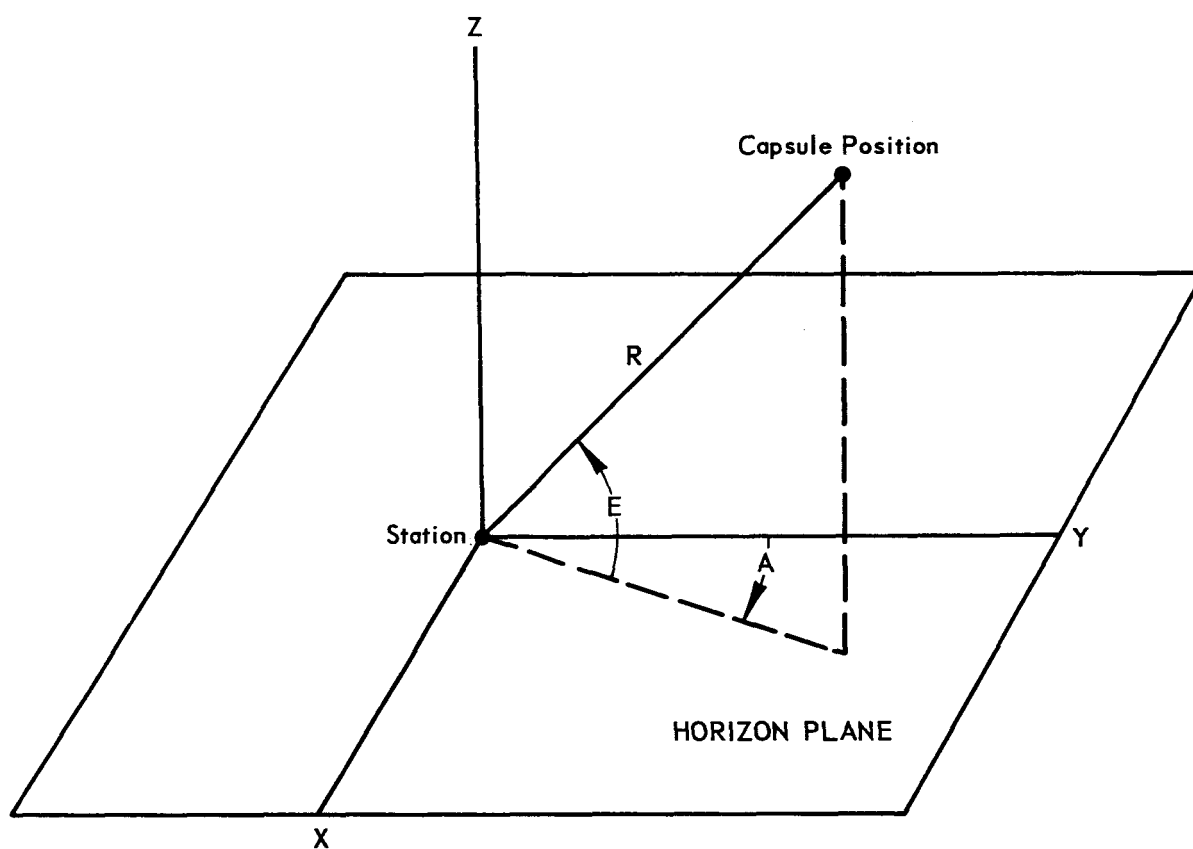
e = Eccentricity (ϕ = angle of eccentricity; $e = \sin \phi$)

ω = Argument of perigee

Π = Longitude of perigee = $\omega + \Omega$

T = Time (usually at perigee or an epoch, i.e., date ~ mean anomaly)

Illustrations (see Figures 4-1 through 4-8) on the following pages depict the coordinate systems; the angular relationships of latitude and longitude; several relative angular values between the earth, orbit and the capsule; and orbital projection factors.



- X axis – Axis in horizon plane perpendicular to Y axis
- Y axis – Axis in horizon plane directed toward north pole
- Z axis – Axis perpendicular to earth surface; zenith axis
- A – Azimuth angle; measured clockwise from north pole
- E – Elevation Angle
- R – Slant range

FIGURE 4-1. RADAR COORDINATE SYSTEM

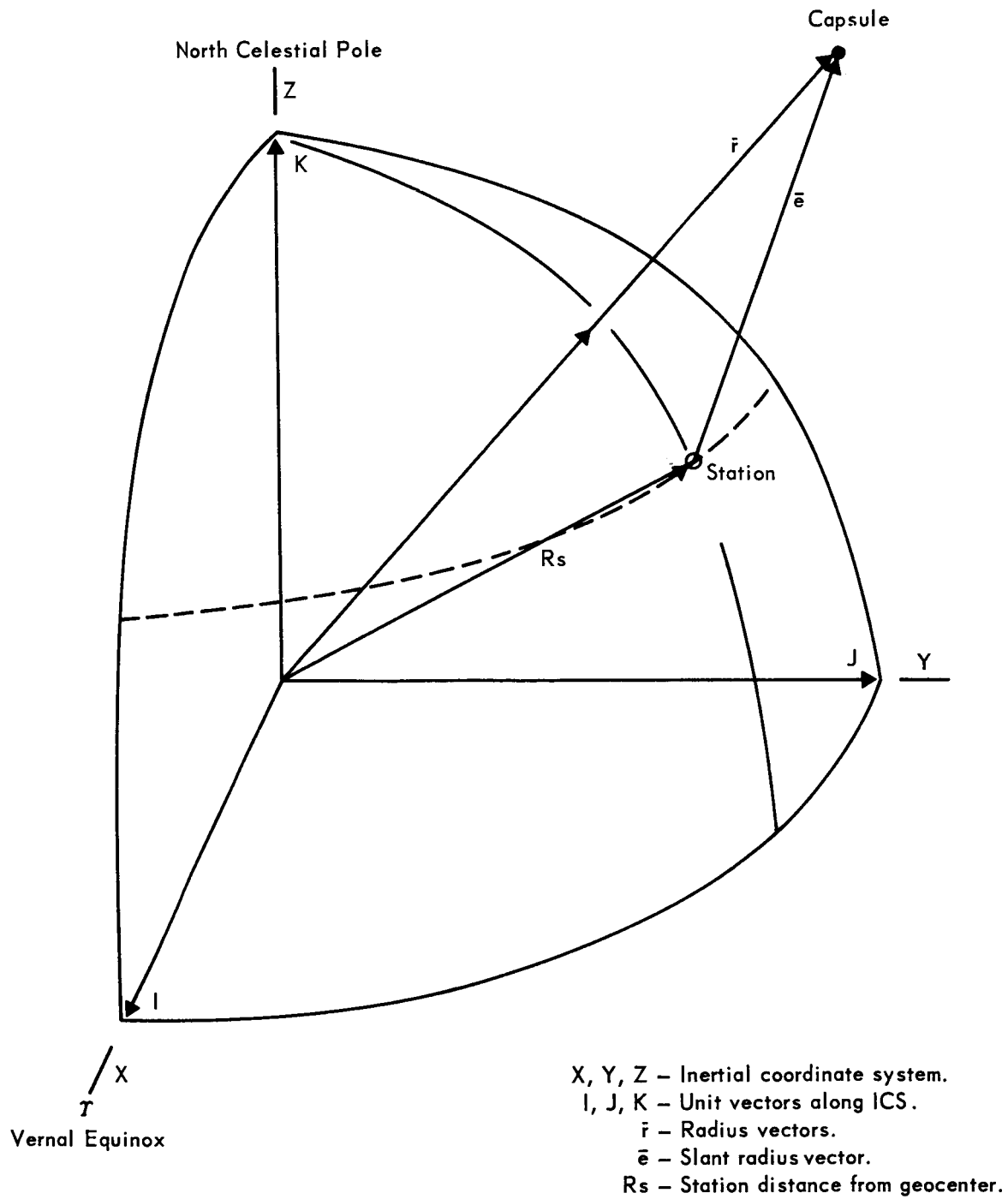
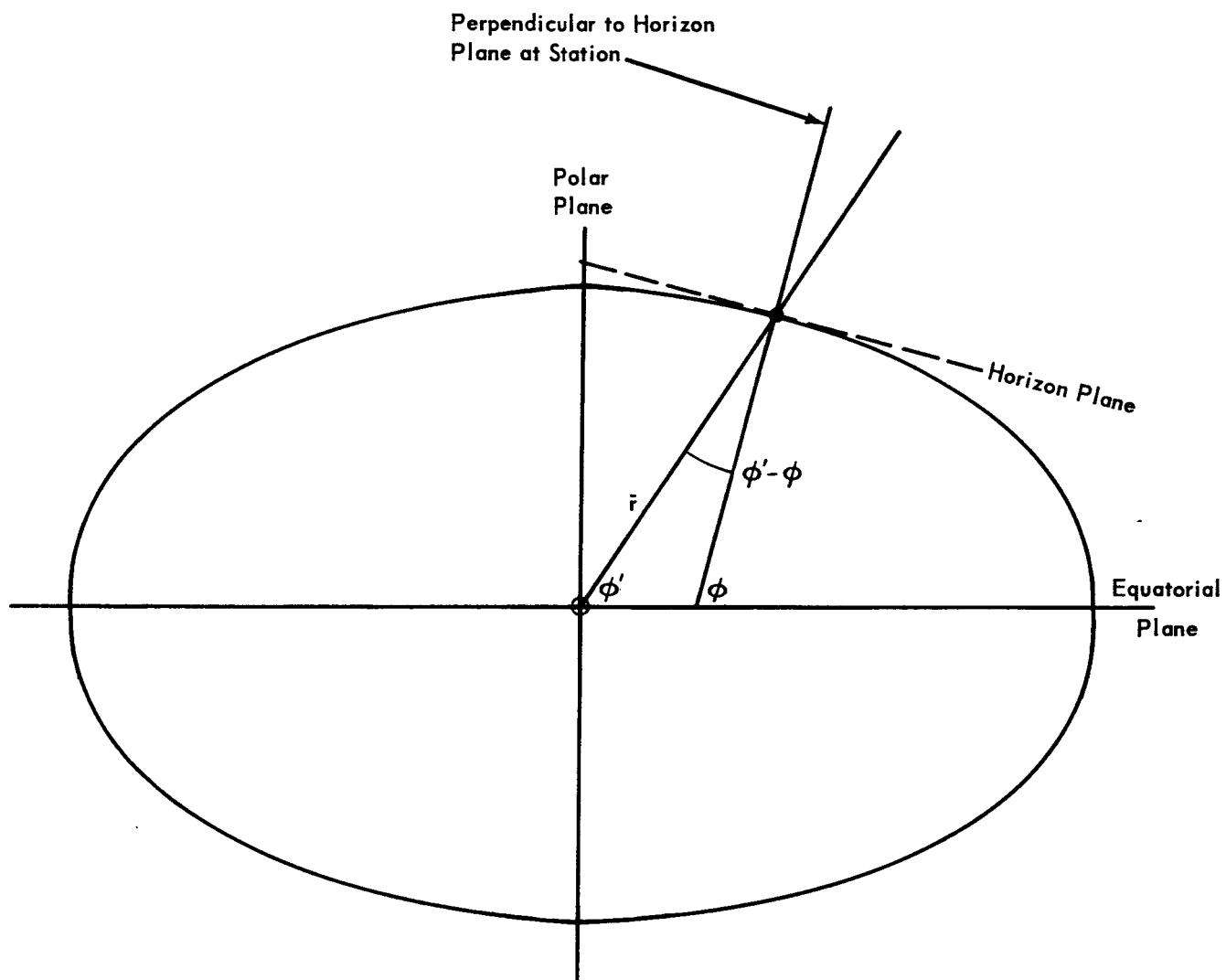


FIGURE 4-2. OBSERVATIONAL FRAMEWORK, INERTIAL COORDINATE SYSTEM



NOTE: Oblateness of the earth is slightly exaggerated.

ϕ' - Geocentric latitude
 ϕ - Geodetic latitude
 \bar{r} - Radius vector

FIGURE 4-3. LATITUDE RELATIONSHIPS

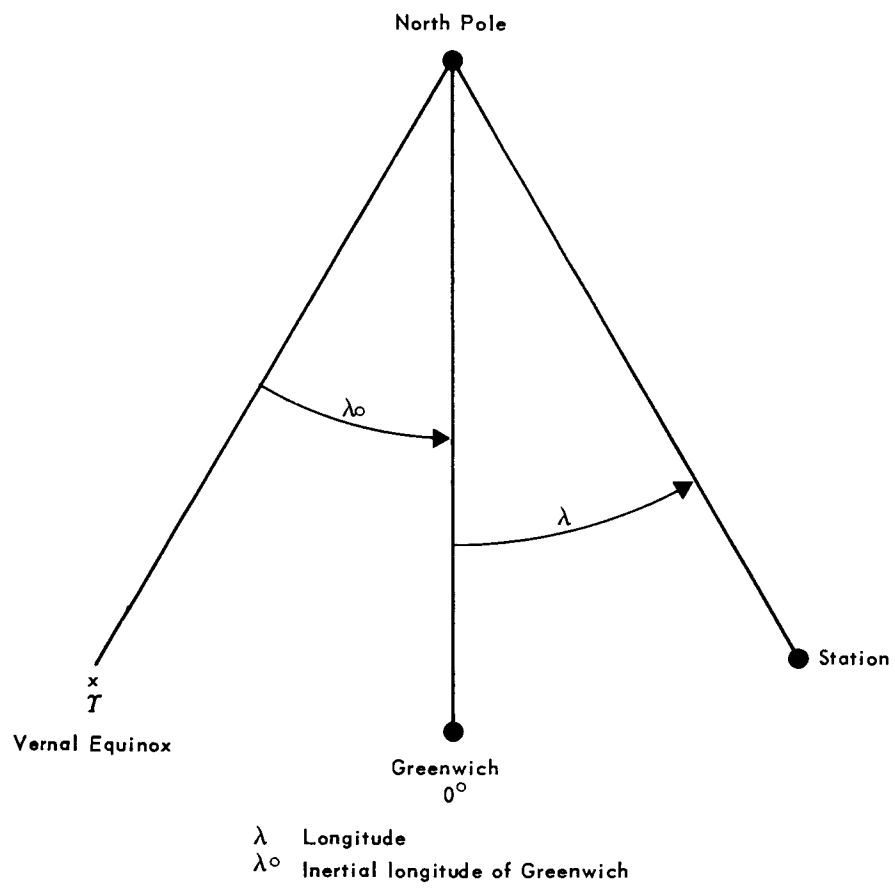


FIGURE 4-4. LONGITUDE RELATIONSHIPS

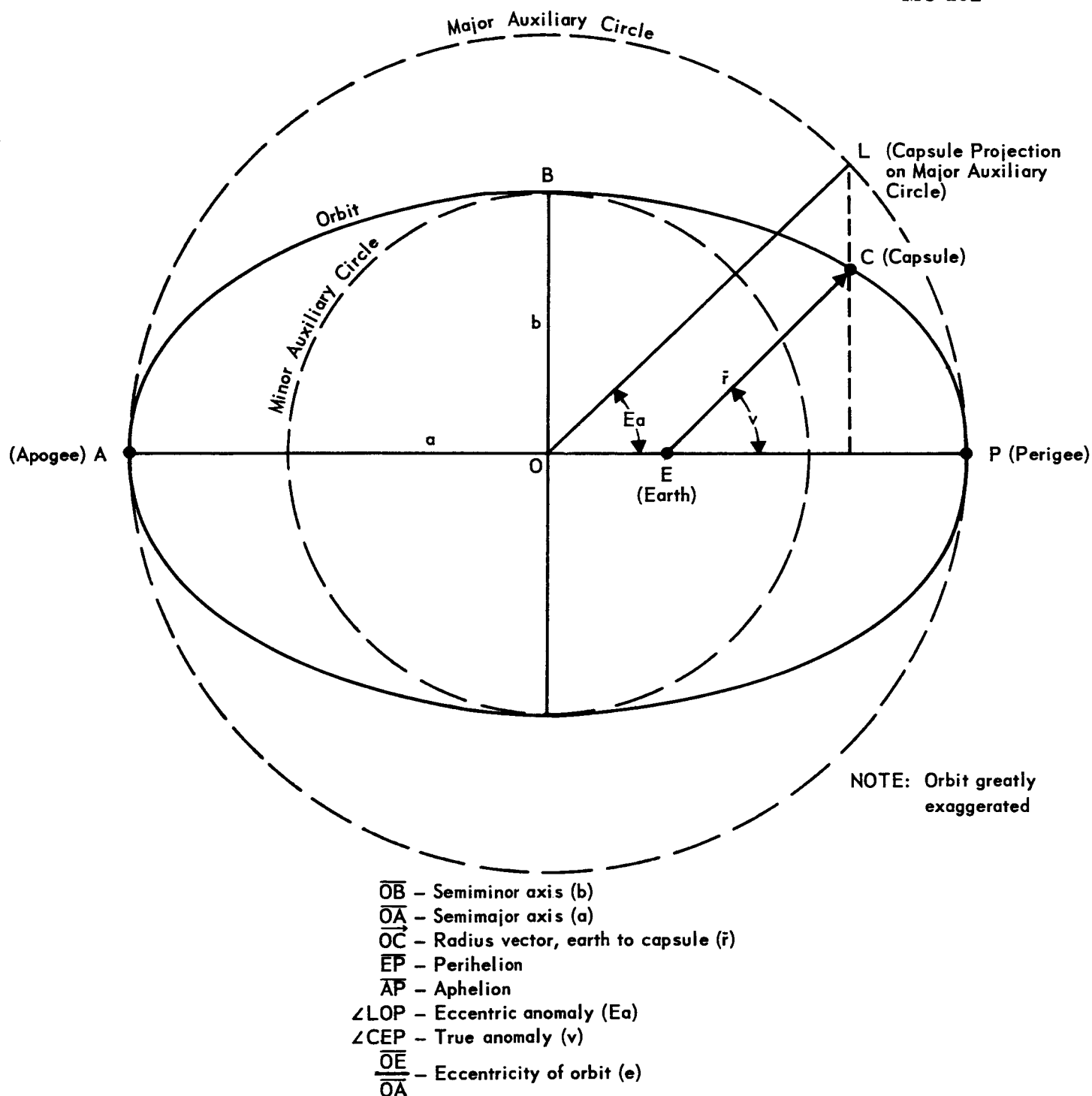
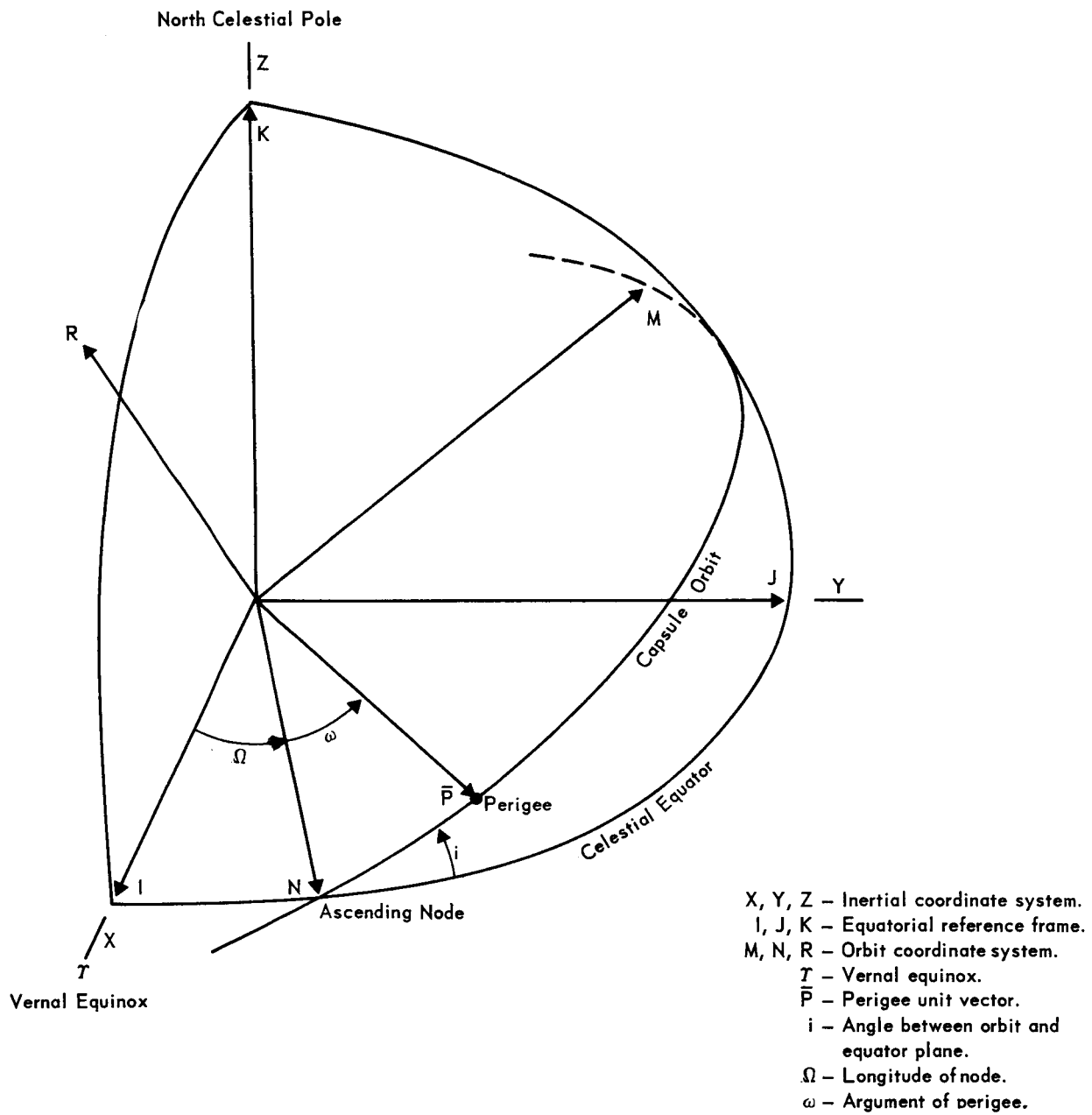


FIGURE 4-5. RELATIONSHIPS BETWEEN EARTH, ORBIT AND CAPSULE



**FIGURE 4-6. PROJECTION OF ORBIT ON CELESTIAL SPHERE
(UNIT VECTORS AND ANGLES DISPLAYED)**

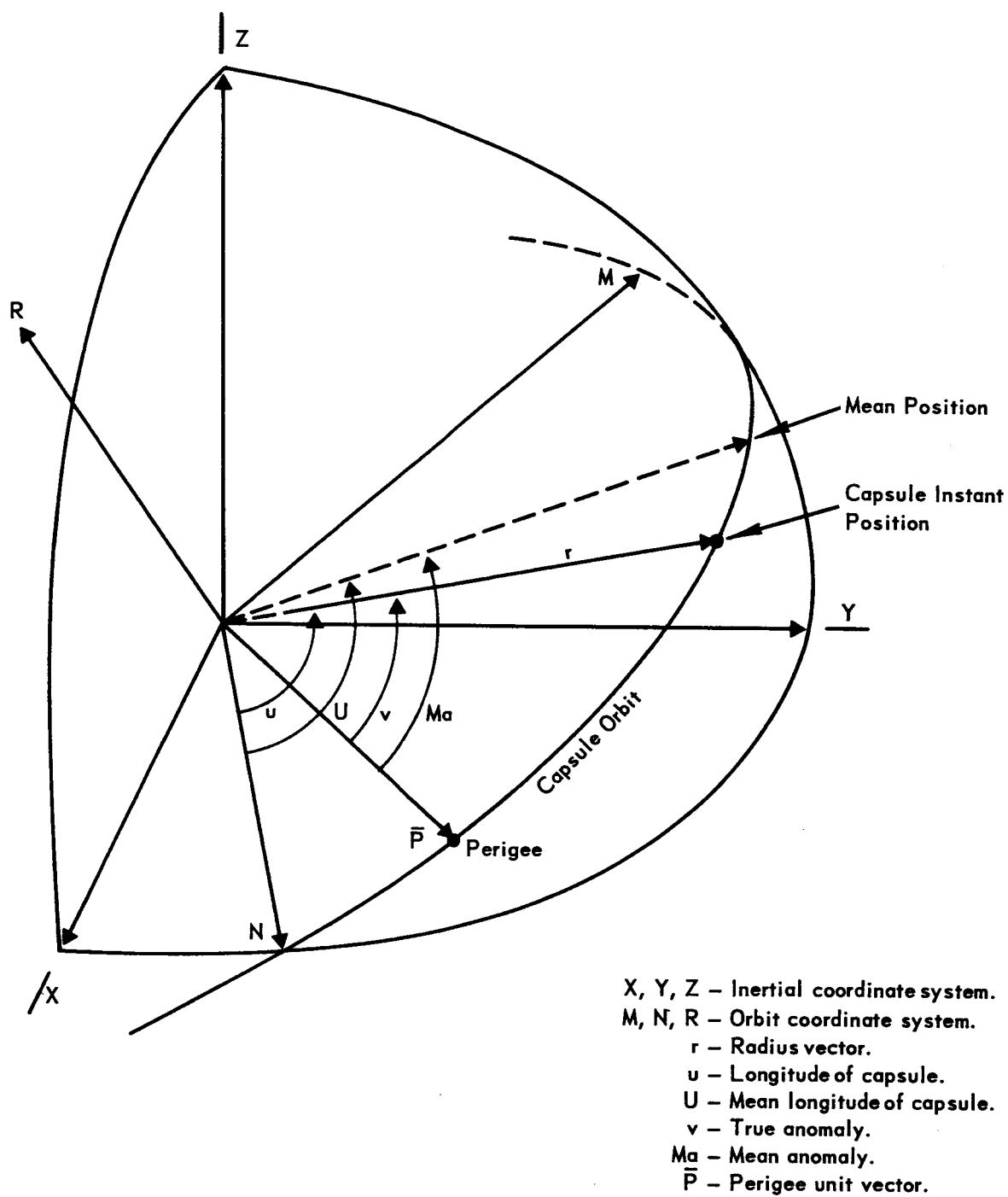


FIGURE 4-7. PROJECTION OF ORBIT ON CELESTIAL SPHERE
(LONGITUDES AND ANOMALIES DISPLAYED)

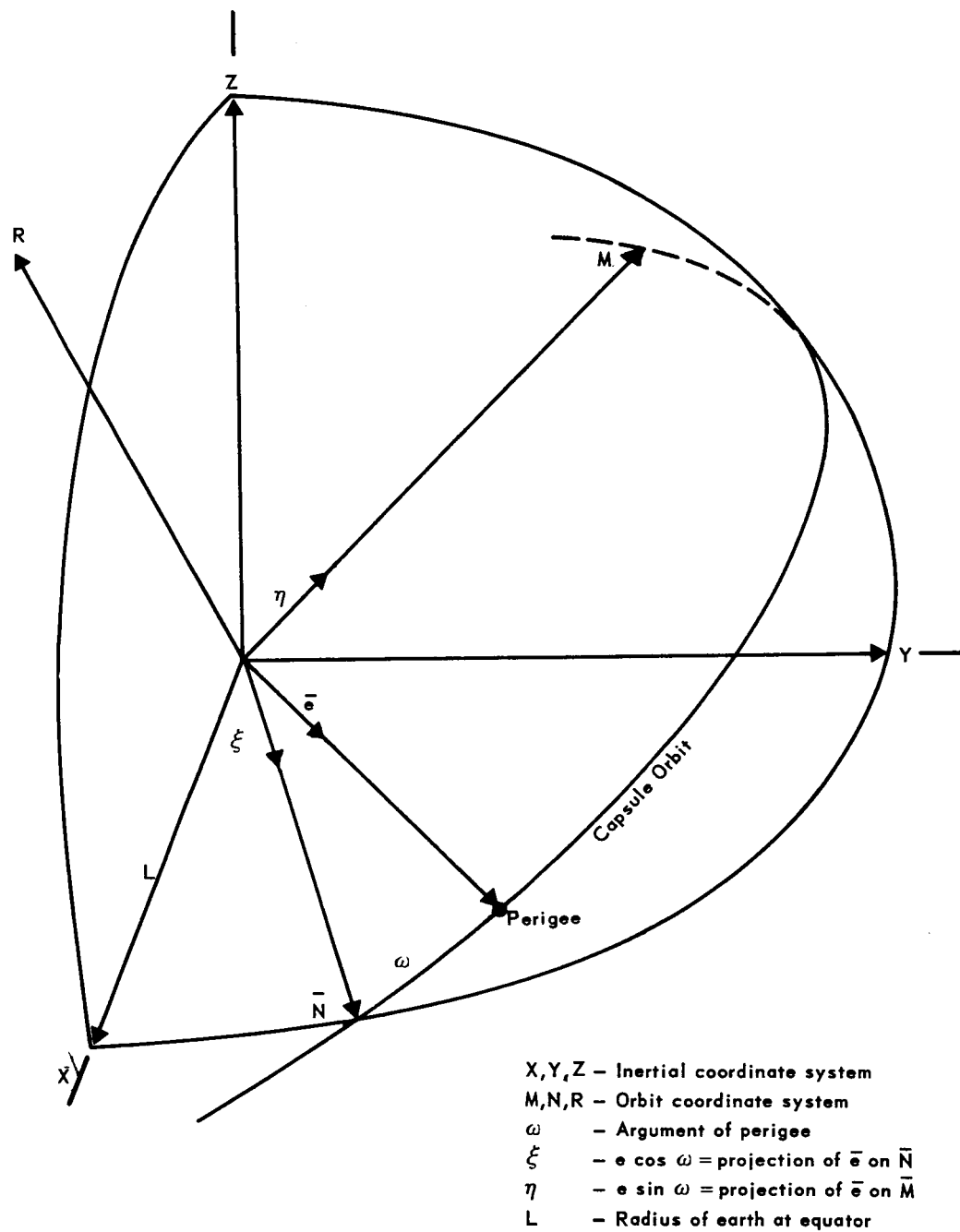


FIGURE 4-8. XI, ETA COORDINATES

4.2.1 National Bureau of Standards Conversion Factors

The conversion factors listed below are constant values which have been established from National Bureau of Standards weights and measures. Participating Project Mercury organizations have agreed on these basic measurements as the reference standards upon which to base pertinent computations.

- a) One international foot = 0.3048 meter (exact).
- b) One international nautical mile = 1852 meters.
- c) One international pound = 0.4535923 kilograms.
- d) One slug = $9.80665 \div 0.3048 = 32.17404855$ pounds (International Commission on Weights and Measures).
- e) One American Survey foot = 0.30480061 meter.

4.2.2 Real Time Impact Prediction Coordinate Transformations

The following paragraphs indicate the coordinate systems, transformations and format of the data to be transmitted to NASA in real time. Position and velocity components, transmitted at 0.2-second intervals are arranged in the following format (in the order shown):

- t = time of data referenced to first motion (counts in units of 0.1 second in four-bit BCD)
- x = parameters in floating binary
- y = inertial position and velocity
- z = components (see Figure 4-11)
- \dot{x} = position in CUD (20,925,672.5 ft.)
- \dot{y} = velocity in CUD (CUD/CUT)
- \dot{z} = CUT = 806.832 sec.
- Σ = logical checksum of t, x, y, z, \dot{x} , \dot{y} , \dot{z}

Local Azusa (Mark I) Coordinate (x'' , y'' , z'') System (see Figure 4-9): This system is a configuration of angles between Azusa tracking system baselines and the local vertical. The $x'' y''$ plane is assumed to be tangent to the earth at the Azusa site; z'' is the local vertical. A clockwise rotation of $7^{\circ} 10' 53'' 6$ about the z'' axis orients the system such that the y' axis is north and the x' axis is east.

Local Radar Coordinate ($x' y' z'$) System (see Figure 4-10): The $x' y'$ plane is tangent to the earth at the radar site, with y' north and x' east; z' is the local vertical.

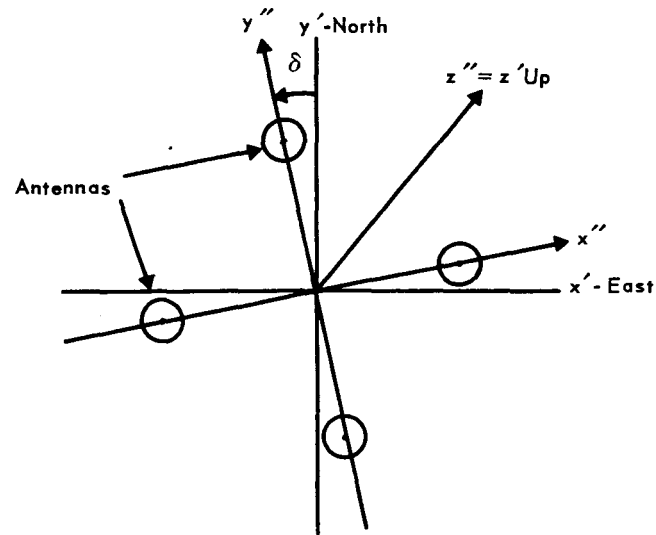


FIGURE 4-9. LOCAL AZUSA (MARK I) COORDINATES

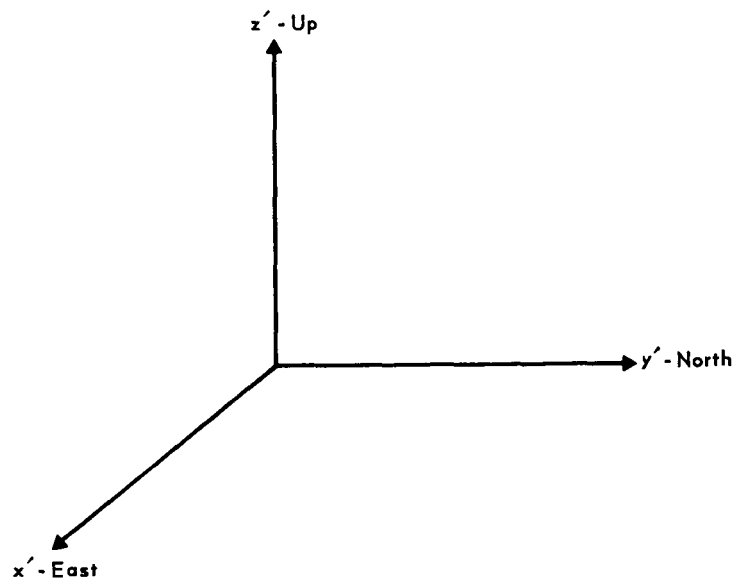


FIGURE 4-10. LOCAL RADAR COORDINATE SYSTEM

Auxiliary Coordinate System ($\hat{x}\hat{y}\hat{z}$)(see Figure 4-11): This system is a moving geocentric equatorial system with its origin at the center of the earth. The \hat{z} axis is the axis of rotation of the earth, the \hat{x} axis is in the plane of the equator through the Greenwich Meridian, and the \hat{y} axis is chosen so the system ($\hat{x}\hat{y}\hat{z}$) is right-handed.

Inertial Coordinate System (x, y, z) (see Figure 4-11): This is the system in which the elliptical trajectory parameters and impact point are computed. The origin is at the center of the earth (assumed unaccelerated for the duration of flight), and the $x y z$ system coincides with $\hat{x} \hat{y} \hat{z}$ at time $t = t_i$; $\hat{x} \hat{y} \hat{z}$ is a moving system, and $x y z$ is assumed fixed with respect to the stars.

Azusa Transformations:

- a) Azusa data is sent to the Impact Predictor computer in the form of two direction cosines (l'_i and m'_i) and a slant range (R'_i). After these parameters are corrected for parallax, zero sets and refraction, they become l_i , m_i and R_i .

- b) The rectangular coordinates in the local Azusa system (x''_i, y''_i, z''_i) are:

$$x''_i = l_i R_i$$

$$y''_i = m_i R_i$$

$$z''_i = R_i (1 - l_i^2 - m_i^2)^{1/2}$$

- c) The rectangular coordinates are rotated to the north/east system (x'_i, y'_i, z'_i) by:

$$x'_i = x''_i \cos \delta - y''_i \sin \delta$$

$$y'_i = x''_i \sin \delta + y''_i \cos \delta$$

$$z'_i = z''_i$$

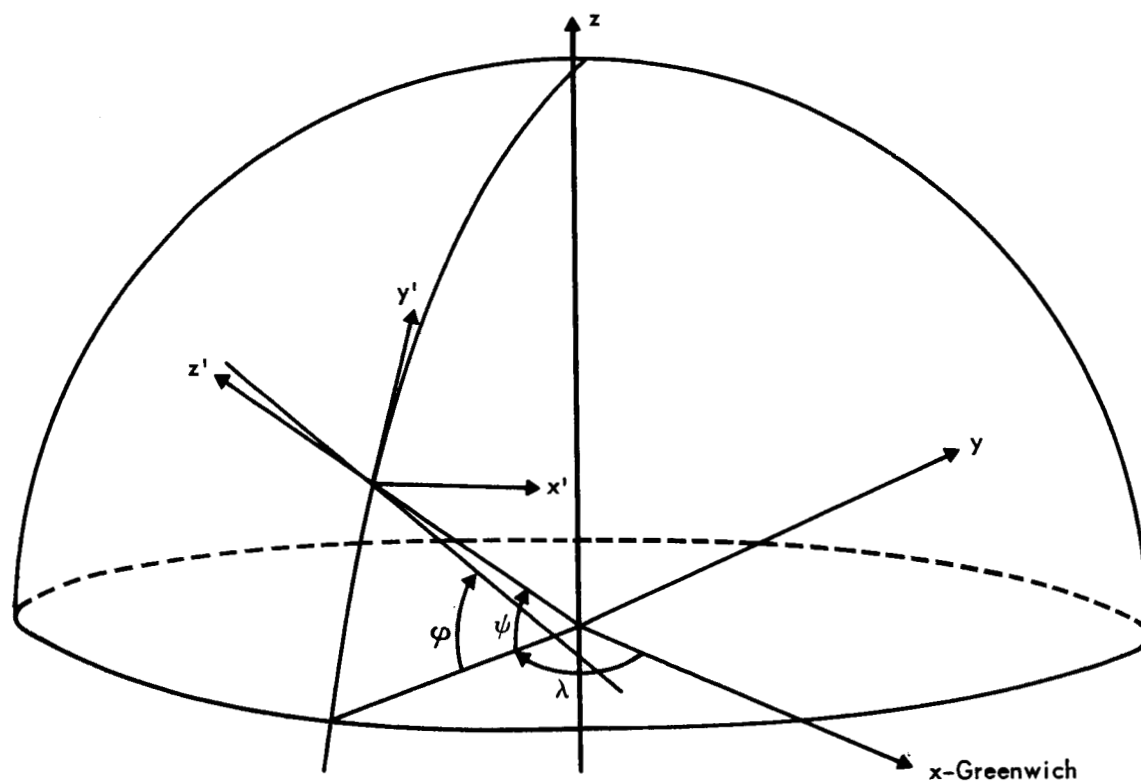
- d) The coordinates are transformed into the inertial system (x, y, z) by:

$$x_i = x_0 + a_{11} x'_i + a_{12} y'_i + a_{13} z'_i$$

$$y_i = y_0 + a_{21} x'_i + a_{22} y'_i + a_{23} z'_i$$

$$z_i = z_0 + a_{31} x'_i + a_{32} y'_i + a_{33} z'_i$$

where a_{ij} values are the direction cosines of the x', y', z' system, and the x_0, y_0, z_0 , are the position components of the Azusa site in the x, y, z system. These are defined by:



φ : Geodetic latitude $-\frac{\pi}{2} \leq \varphi \leq \frac{\pi}{2}$

ψ : Geocentric latitude $-\frac{\pi}{2} \leq \psi \leq \frac{\pi}{2}$

λ : Longitude (positive west) $-\pi \leq \lambda \leq \pi$

FIGURE 4-11. INERTIAL COORDINATE SYSTEM

$$\begin{aligned}
a_{11} &= \sin \lambda & x_0 &= R_0 \cos \psi \cos \lambda + h a_{13} \\
a_{21} &= \cos \lambda & y_0 &= R_0 \cos \psi \sin \lambda + h a_{23} \\
a_{31} &= 0 & z_0 &= R_0 \sin \psi + h a_{23} \\
a_{12} &= -\cos \lambda \sin \varphi \\
a_{22} &= \sin \lambda \sin \varphi & \psi &= \tan^{-1} \left[\frac{c^2}{a^2} \tan \varphi \right] \\
a_{32} &= \cos \varphi \\
a_{13} &= \cos \lambda \cos \varphi \\
a_{23} &= -\sin \lambda \cos \varphi & R_0 &= \frac{ac}{[(a^2 - c^2) \sin^2 \psi + c^2]^{\frac{1}{2}}} \\
a_{23} &= \sin \varphi
\end{aligned}$$

where:

$a = 20925832$ ft. = semimajor axis

$c = 20854892$ ft. = semiminor axis

based on the Clarke spheroid of 1866

h = height of Azusa site.

- e) Azusa data (l_i , m_i , R_i) is smoothed to obtain velocity components (\dot{l} , \dot{m} , \dot{R}) by:

$$\begin{aligned}
\dot{l}_i &= c_1 \sum_j a_j l_{i+j} \\
\dot{m}_i &= c_1 \sum_j a_j m_{i+j} & j &= -16 \rightarrow 4 \\
\dot{R}_i &= c_1 \sum_j a_j R_{i+j}
\end{aligned}$$

where the a_i values are least-squares-determined data multipliers.

- f) Velocity components in the local Azusa system (x' , y' , z') are:

$$\begin{aligned}
\dot{x}'_i &= R_i \dot{l}_i + \dot{R}_i l_i \\
\dot{y}'_i &= R_i \dot{m}_i + \dot{R}_i m_i \\
\dot{z}'_i &= \frac{1}{z'_i} (R_i \dot{R}_i - x'_i \dot{x}'_i - y'_i \dot{y}'_i)
\end{aligned}$$

g) The components are rotated into the north/east system (x, y, z) by:

$$\dot{x}_i' = \dot{x}_i' \cos \delta - \dot{y}_i' \sin \delta$$

$$\dot{y}_i' = \dot{x}_i' \sin \delta + \dot{y}_i' \cos \delta$$

$$\dot{z}_i' = \dot{z}_i'$$

h) The rotation of components into the geocentric equatorial system is:

$$\hat{x}_i = a_{11}\dot{x}_i' + a_{12}\dot{y}_i' + a_{13}\dot{z}_i'$$

$$\hat{y}_i = a_{21}\dot{x}_i' + a_{22}\dot{y}_i' + a_{23}\dot{z}_i'$$

$$\hat{z}_i = a_{31}\dot{x}_i' + a_{32}\dot{y}_i' + a_{33}\dot{z}_i'$$

i) Inertial velocity components are then:

$$\dot{x}_i = \hat{x}_i - \omega y_i$$

$$\dot{y}_i = \hat{y}_i + \omega x_i$$

$$\dot{z}_i = \hat{z}_i$$

where ω is the angular velocity of the earth.

$$(\omega = .058835124 \text{ rad/CUT})$$

Radar:

a) Local radar position (x' y' z'):

$$x_i' = R_i \cos E_i \sin A_i$$

$$y_i' = R_i \cos E_i \cos A_i$$

$$z_i' = R_i \sin E_i$$

b) Position in the inertial system (x, y, z) is obtained by:

$$x_i = x_0 + a_{11}x_i' + a_{12}y_i' + a_{13}z_i'$$

$$y_i = y_0 + a_{21}x_i' + a_{22}y_i' + a_{23}z_i'$$

$$z_i = z_0 + a_{31}x_i' + a_{32}y_i' + a_{33}z_i'$$

where the a_{ij} values are the direction cosines of the local radar axes (x, y, z) with respect to the x, y, z system, and x_0, y_0, z_0 are the position components of the radar site in the x, y, z system.

- c) Radar data (R, A, E) is smoothed to obtain velocity components (\dot{R} , \dot{A} , \dot{E}) by:

$$\dot{A}_i = C_2 \sum_j \beta_j A_{i+j}$$

$$\dot{E}_i = C_2 \sum_j \beta_j E_{i+j} \quad j = -46 \rightarrow 4$$

$$\dot{R}_i = C_2 \sum_j \beta_j R_{i+j}$$

where the β_j values are least-squares-determined data multipliers.

- d) Velocity components (x' , y' , z') in the local radar system are:

$$\dot{x}'_i = \dot{R}_i \cos E_i \sin A_i - \dot{E}_i R_i \sin E_i \sin A_i + A_i y'_i$$

$$\dot{y}'_i = \dot{R}_i \cos E_i \cos A_i - \dot{E}_i R_i \sin E_i \cos A_i - A_i x'_i$$

$$\dot{z}'_i = \dot{R}_i \sin E_i + \dot{E}_i R_i \cos E_i$$

- e) The components are further transformed into a geocentric equatorial system by:

$$\hat{x}_i = a_{11}\dot{x}'_i + a_{12}\dot{y}'_i + a_{13}\dot{z}'_i$$

$$\hat{y}_i = a_{21}\dot{x}'_i + a_{22}\dot{y}'_i + a_{23}\dot{z}'_i$$

$$\hat{z}_i = a_{31}\dot{x}'_i + a_{32}\dot{y}'_i + a_{33}\dot{z}'_i$$

- f) The inertial velocity components are:

$$\dot{x}_i = \hat{x}_i - \omega y_i$$

$$\dot{y}_i = \hat{y}_i + \omega x_i$$

$$\dot{z}_i = \hat{z}_i$$

Transformation Constants:

The following values are the rotation and translation constants associated with present Cape Canaveral instrumentation.

	1.16	XN-1	XN-2
φ	28°28'52''.792	28°13'35''.279	26°36'54''.984
λ	80°34'36''.231	80°35'58''.051	78°20'53''.188
h	44.79 ft.	27.12 ft.	46.38 ft.
a_{11}	.98650575	.98657061	.97939272
a_{21}	.16372662	.16333529	.20196503
a_{31}	0	0	0
a_{12}	-.07807670	-.07725070	-.09047981
a_{22}	.47043734	.46660627	.43876541
a_{32}	.87897253	.88108507	.89403484
a_{13}	.14391120	.14391229	.18056378
a_{23}	-.86711145	-.86925263	-.87561122
a_{33}	.47687238	.47295780	.44799742
x_0	3013788.4	3013770.6	3781024.7
y_0	-18159048.0	-18203644.0	-18335392.0
z_0	9919065.3	9837508.8	9317616.3

	5.16	Azusa MK I
φ	24°7'6''.114	28°29'28''.9451
λ	74°30'15''.83	80°33'32''.2744
h	42.91 ft.	21.9 ft
a_{11}	.96365096	.98645493
a_{21}	.26716442	.16403250
a_{31}	0	0
a_{12}	-.10916953	-.07824784
a_{22}	.39376997	.47056508
a_{32}	.91270324	.87888893
a_{13}	.24384183	.14416634
a_{23}	-.87952735	-.86698432
a_{33}	.40862302	.47702643
x_0	5105489.5	3019129.7
y_0	-18415288.0	-18156374.0
z_0	8497720.5	9922263.3

4.3 CONSTANTS

The letter K is the first prefix character in all constants; the five characters following it may be mnemonic. An effort has been made to assign names to Project Mercury constants in such a way that the definition and forms of the constant are evident from the symbol itself.

Quantities within the system are named according to adopted terminology. Integers to 99999 which are in the address are assigned their actual values; if the value consists of less than five integers, it is preceded by zeroes. The letter D is the second character of those integers in the decrement. Floating point constants include decimal points, wherever possible.

Four types of constants are indicated on the following pages: alphabetical, numerical, octal and physical. The alphabetical and numerical listings (Subsections 4.3.1 and 4.3.2, respectively) contain identical values merely arranged in different order. Though present in both listings, physical constants are indicated by asterisks only in the alphabetical listings. Octal constants are presented separately in Subsection 4.3.3.

The numerical listing pro forma is not accepted by the computer; however, alphabetical and octal listings are accepted. Physical constants differ from numerical constants in that the former are generally measurements and the latter are not.

4.3.1 Constants (Alphabetical Order)

K00000	DEC	0
K00001	DEC	1
K00002	DEC	2
K00003	DEC	3
K00004	DEC	4
K00005	DEC	5
K00006	DEC	6
K00007	DEC	7
K00008	DEC	8
K00009	DEC	9
K00010	DEC	10
K00011	DEC	11
K00012	DEC	12
K00013	DEC	13
K00015	DEC	15
K00016	DEC	16
K00017	DEC	17
K00018	DEC	18
K00019	DEC	19
K00020	DEC	20
K00021	DEC	21
K00023	DEC	23
K00024	DEC	24
K00025	PZE	25
K00026	DEC	26
K00027	DEC	27
K00028	DEC	28
K00030	DEC	30

IOMANI

MC 102

K00031 DEC	31	
K00032 DEC	32	
K00033 DEC	33	
K00034 DEC	34	
K00035 DEC	35	
K00037 DEC	37	
K00039 DEC	39	
K00040 DEC	40	
K00041 DEC	41	
K00042 DEC	42	
K00045 DEC	45	
K00046 DEC	46	
K00047 DEC	47	
K00048 DEC	48	
K00051 DEC	51	
K00052 DEC	52	
K00053 DEC	53	
K00054 DEC	54	
K0005. DEC	5.	
K00060 DEC	60	
K00063 DEC	63	IOMANI
K00066 DEC	66	
K00075 DEC	75	
K00084 DEC	84	MYSCRD
K00085 DEC	85	
K00086 DEC	86	
K00090 DEC	90	
K000.1 DEC	.1	
K000.2 DEC	.2	
K000.5 DEC	0.5	
K000.9 DEC	.900	
K000JB PZE	30	NON CONSTANT
K000NB PZE	30	
K00120 DEC	120	
K00154 DEC	154	
K00155 DEC	155	
K00156 DEC	156	
K00160 DEC	160	
K00163 DEC	163	
K00164 DEC	164	
K00165 DEC	165	
K00166 DEC	166	
K00167 DEC	167	
K00168 DEC	168	
K00179 DEC	179	
K00180 DEC	180	
K00198 DEC	198	
K001.0 DEC	1.0	
K001.4 DEC	1.4	
K001.5 DEC	1.50	
K001S1 PZE		
K00218 DEC	218	
K00219 DEC	219	
K00220 DEC	220	
K00221 DEC	221	
K00222 DEC	222	
K00224 DEC	224	
K00256 DEC	256	
K01800 DEC	1800	
K002.0 DEC	2.0	
K002.5 DEC	2.5	
K003.0 DEC	3.0	
K00420 DEC	420	
K004.0 DEC	4.	

ZP8 - L,A
 NO. OF ATTEMPTS TO CONVERGE TABLE
 NO. OF ATTEMPTS TO CONVERGE ONCE SET

IF ONE,MUTILATE DENSITY, K ONCE SET N4DENS

K10 - L,A,O,R

AOSTAD

K00512 DEC	512	K10P - L,A,O,R	
K005.0 DEC	5.0		
K0060. DEC	60.		
K006.0 DEC	6.0		
K00.01 DEC	.01	-ZP2-L,A	
K00.04 DEC	.04	-ZP2 - L,A	
K00.25 DEC	0.25		
* K00.A1 DEC	.46038333		
* K00.A2 DEC	-1.09306667		
* K00.A3 DEC	1.41426667		
* K00.A4 DEC	-1.0424		
* K00.A5 DEC	.41235		
* K00.A6 DEC	-.0682		
* K00.B0 DEC	.065495756		
* K00.B1 DEC	.067409063		
* K00.B2 DEC	-.110635752		
* K00.B3 DEC	.104370596		
* K00.B4 DEC	-.05999091		
* K00.B5 DEC	.019393189		
* K00.B6 DEC	-.002708609		
* K00.C2 DEC	.00324696		
* K00.G0 DEC	-.0963348773		
* K00.G1 DEC	.0074115416		
* K00.G2 DEC	-.0009904103		
* K00.G3 DEC	.0000796407		
* K00.H0 DEC	-.5		
* K00.H1 DEC	.0648396164		
* K00.H2 DEC	-.0139550265		
* K00.H3 DEC	.0015790344		
* K00.MH DEC	.074366916		
* K00.PI DEC	3.14159265		
* K00.SH DEC	.0012394486		
* K00.UT DEC	806.8104		
K01000 DEC	1000		
K01023 DEC	1023	K3P - L,A,O,R	
K01024 DEC	1024	K12 - L,A	
K010.0 DEC	10.		
K0130. DEC	130.	-ZP7 - L,A	
* K013.4 DEC	13.44684	MINUTES PER HG UNIT OF TIME	R5RARF
K0150. DEC	150.		
K01.75 DEC	1.75		
K01.E3 DEC	1000.0		
K0300. DEC	300.		
* K03HPI DEC	4.712388975	3 HALVES PI(RADIANS)	
K0400. DEC	400.	-ZP12-O,R	
* K045.4 DEC	.021504685	450,000 FT IN HG UNITS	N4DRAG
* K04.M5 DEC	.0000434146436	CONVERTS FROM KG/M CUBED TO HG UNITS	N4DENS
K0600. DEC	600.		
K07200 DEC	7200		
K0.144 DEC	.144	-ZP11-O,R	
* K0.1DG DEC	.0174532925	ONE DEGREE IN RADIANS	A0STAD
K0.341 DEC	.341	LPBD SCALE FACTOR FOR ORDINATE	
K0.550 DEC	.550		I0HS09
* K0.5DG DEC	.0872664626	-ZP1 - L,A	
K0.650 DEC	.650		I0HSGB
* K0.6E4 DEC	.0028672914	60,000 FT IN HG UNITS	N4DRAG
K0.9E4 DEC	90000.0		
* K0.A1P DEC	2.28573082		
* K0.A2P DEC	-5.1099041		
* K0.A3P DEC	6.55591931		
* K0.A4P DEC	-4.82397487		
* K0.A5P DEC	1.90782077		
* K0.A6P DEC	-.31559193		
* K0.ALT DEC	.00333874		

MC 102

*K0.B0P DEC	.30422454		
*K0.B1P DEC	.46038360		
*K0.B2P DEC	-.54653605		
*K0.B3P DEC	.47142857		
*K0.B4P DEC	-.26060681		
*K0.B5P DEC	.08247355		
*K0.B6P DEC	-.01136740		
K0.CTB DEC	1E-15	TEST IF F TABLE CONVERGED	ASSUMED
*K0.MPR DEC	3437.746771	-DGHR	
*K0.RAD DEC	6378145.	EQUATORIAL RADIUS OF EARTH IN METERS	N4DRAG
*K0.S1D DEC	.01745241	SINE OF ONE DEGREE	A0STAD
*K0.SF1 DEC	5861.358244	SF1 - L,A	
*K0.SF2 DEC	12787.5	SF2 - L,A	
*K0.SF3 DEC	732.6697805	SF3 - L,A,O,R	
*K0.SF4 DEC	162.8155068	SF4 - L,A,O,R	
*K0.SF5 DEC	2442.232602	SF5 - L,A,O,R	
*K0.SF6 DEC	651.2620273	SF6 - L,A,O,R	
KOMIN3 DEC	-3		IOMANI
KOMINS DEC	-3		
KOZERO DEC	0		
K10.E5 DEC	100000.0		
K15360 DEC	15360	-DGHR	
K1.525 DEC	1.525		
K2700. DEC	2700.		
K32.E3 DEC	32000.0		
*K6(G). DEC	-.33189710853		R6BOTH
*K6(H). DEC	-.22386742572		R6BOTH
*K806.8 DEC	806.8134	SECONDS PER IG UNIT OF TIME	POSWIP
*K8.3M3 DEC	8.3E-3	8.3 MILLI SECONDS TO SECONDS	LAUNCH
*KC.NVF DEC	6378.145	KM/HG UNITS OF LENGTH	N4DRAG
*KCD.CV DEC	336939173.2		
*KCD.CW DEC	381660578.3		
*KCFTSC DEC	25936.294946	CONVERSION FACTOR TO FT/SEC	POSWIP
KCNVRG DEC	0.000017	CONVERGENCE CRITERION	
*KE.EEE DEC	2.718281828		
KECR.T DEC	0.003	TEST FOR E CRITICAL	C9RVTH
KFEEBD DEC	2.0E-7		
*KGAMMN DEC	-4.211550	GAMMA MEAN IN DEGREES	POSWIP
*K.016B DEC	.0166666666660	HS INPUT TIME CONV CONSTANT	IOHSGB
*K.12DG DEC	-.2094395102	-ZP5 - L,A,O,R	
*K.1RAD DEC	57.29577951	- L,A,O,R	
*K.2030 DEC	.20302247	LN(SEA LEVEL VALUE OF DENSITY,METRIC)	N4DENS
*K.30SC DEC	.1454441043E-3	- L,A,O,R	
K.40DG DEC	.6981317008	-ZP3 - L,A,O,R	
K.82DG DEC	1.431169987		
*K.BETA DEC	.59341193	ORIENTATION ANGLE OF DELTA V VECTOR	
*K.DELT DEC	.52	FL PT DIFF BET BURNOUT AND NEXT SEC	
*K.DLTA DEC	.00066451326	HG TIME BET BURNOUT + NEXT WHOLE SEC	
*K.ECC2 DEC	6.6934215E-3	ECCENTRICITY OF MERC SPHEROID SQUARED	
*K.INB2 DEC	1.00673852	INVERSE OF MERC SPHEROID SEMIMINOR AXIS SQUARED	
*K.MACH DEC	7905.3827	HG UNITS OF LENGTH/HG UNITS OF TIME	N4COEF
*K.MUTE DEC	1.0	DENSITY MUTILATION COEFFICIENT	N4DENS
*K.N4HC DEC	136025.0	CRITICAL ALTITUDE DENSITY FORMULA	N4PDEN
*K.OMEG DEC	.058833543	ROTATIONAL VELOC OF EARTH (HG UNITS)	N4DRAG
*K.OMES DEC	7.29211508E-5	ROTATION OF EARTH IN RAD/SEC	
*K.RTER DEC	.00437526905	ROTATION OF THE EARTH	A0STAD
K.TCTB DEC	1E-15	TEST IF F CONVERGED	ASSUMED VALUE
*KINTV2 DEC	2000	STORE EVERY 2000TH INTEGRATION STEP	R5RARF
*KKK2PI DEC	6.283185307		
*KKKPI2 DEC	1.57079632675	HALF PI IN RADIANS	POSWIP
*KL.HGT DEC	141766.0		
*KLPBSF DEC	5.24615384	LPBD SCALE FACTOR FOR ABSCISSA	
KM0000 DEC	-0		
KM0001 DEC	-1		

KM01.0 DEC	-1.0		
*KMU.80 DEC	.02538390	MERCURY TIME OF BURNOUT	R3RVB0
*KMUYDS DEC	6975224.19	NUMBER OF YARDS IN ONE MERCURY UNIT	A0STAD
*KS.LNG DEC	160.41646		
*KS.UND DEC	20.0463333	PROPORTIONALITY CONSTANT--CS-TM EQ	N4COEF
*KT.GHT DEC	172205.0		
*KT.MPE DEC	288.16	BASE MOLECULAR SCALE TEMP IN DEG K	N4COEF
*KTFLMN DEC	273.579971	TIME OF FALL MEAN IN SECONDS	POSWIP
*KTHEMN DEC	13.209693	THETA MEAN IN DEGREES	POSWIP
*KVD.OR DEC	.017504469	VELOCITY 454 FT PER SEC HG UNITS	
*KVELMN DEC	23306.4968	VELOCITY MEAN IN FT/SECOND	POSWIP

4.3.2 Constants (Numerical Order)

K001SI PZE		IF ONE,MUTILATE DENSITY,K ONCE SET	N4DENS
KM0000 DEC	-0000		
K00.G2 DEC	-0000.0009904103		
K00.B6 DEC	-0000.002708609		
K0.B6P DEC	-0000.01136740		
K00.H2 DEC	-0000.0139550265		
K00.B4 DEC	-0000.05999091		
K00.A6 DEC	-0000.0682		
K00.G0 DEC	-0000.0963348773		
K00.B2 DEC	-0000.110635752		
K6(H). DEC	-0000.1988341488		
K.12DG DEC	-0000.2094395102	-ZP5 - L,A,O,R	
K0.B4P DEC	-0000.26060681		
K6(G). DEC	-0000.2947837748		
K0.A6P DEC	-0000.31559193		
K00.H0 DEC	-0000.5		
K0.B2P DEC	-0000.54653605		
KM0001 DEC	-0001		
KM01.U DEC	-0001.		
K00.A4 DEC	-0001.0424		
K00.A2 DEC	-0001.09306667		
K0MIN3 DEC	-0003		
KGAMMN DEC	-0004.211550	GAMMA MEAN IN DEGREES	IOMANI
K0.A4P DEC	-0004.82397487		POSWIP
K0.A2P DEC	-0005.1099041		
K00000 DEC	00000		
K0ZERO DEC	00000		
K0.CTB DEC	1E-15	TEST IF F TABLE CONVERGED ASSUMED	
K.TCTB DEC	1E-15	TEST IF F CONVERGED ASSUMED VALUE	
KFEEDB DEC	2.0E-7		
KCNVRG DEC	00000.000017	CONVERSION CRITERION	
K04.M5 DEC	00000.0000434146436	CONVERTS FROM KG/M CUBED TO HG UNITS	N4DENS
K.OMES DEC	00000.0000729211508	ROTATION OF EARTH IN RAD/SEC	
K00.G3 DEC	00000.0000796407		
K.DLTA DEC	00000.00066451326		
K00.SH DEC	00000.0012394486		
K00.H3 DEC	00000.0015790344		
K0.6E4 DEC	00000.0028672914	60,000 FT IN HG UNITS	N4DRAG
KECR.T DEC	00000.003	TEST FOR E CRITICAL	C9RVTH
K00.C2 DEC	00000.00324696		
K0.ALT DEC	00000.00333874		
K.RTER DEC	00000.00437526905	ROTATION OF THE EARTH	A0STAD
K.ECC2 DEC	00000.0066934215	ECCENTRICITY OF MERC.SPHEROID SQUARED	
K00.G1 DEC	00000.0074115416		
K8.3M3 DEC	00000.0083	8.3 MILLISECONDS TO SECONDS	LAUNCH
K00.01 DEC	00000.01	-ZP2-L,A	
K.016B DEC	00000.0166666666680	HS INPUT TIME CONVERSION CONSTANT	IOH5GB
K0.S1D DEC	00000.01745241	SINE OF ONE DEGREE	A0STAD

MS 102

K0.1DG DEC	00000.0174532925	ONE DEGREE IN RADIANS	A0STAD
KVD.0R DEC	00000.017504469	VELOCITY 454 FT PER SEC HG UNITS	
K00.B5 DEC	00000.019393189		
K045.4 DEC	00000.021504685	450,000 FT IN HG UNITS	N4DRAG
KMU.80 DEC	00000.02538390	MERCURY TIME OF BURNOUT	R3RVEO
K00.04 DEC	00000.04	-ZP2 ~ L,A	
K.OMEG DEC	00000.058833543	ROTATIONAL VELOC OF EARTH (HG UNITS)	N4DRAG
K00.H1 DEC	00000.0648396164		
K00.80 DEC	00000.065495756		
K00.B1 DEC	00000.067409063		
K00.MH DEC	00000.074366916		
K0.B5P DEC	00000.08247355		
K0.5DG DEC	00000.0872664626	-ZP1 ~ L,A	
K000.1 DEC	00000.1		
K00.B3 DEC	00000.104370596		
K0.144 DEC	00000.144	-ZP11-0,R	
K.30SC DEC	00000.1454441043E-3	-L,A,O,R	
K000.2 DEC	00000.2		
K.2030 DEC	00000.20302247	LN(SEA LEVEL VALUE OF DENSITY,METRIC)	N4DENS
K00.25 DEC	00000.25		
K0.B0P DEC	00000.30422454		
K0.341 DEC	00000.341		
K00.A5 DEC	00000.41235		
K00.A1 DEC	00000.46038333		
K0.B1P DEC	00000.46038360		
K0.B3P DEC	00000.47142857		
K000.5 DEC	00000.5		
K.DELT DEC	00000.52	FL PT DIFF BET BURNOUT AND NEXT SEC	
K0.550 DEC	00000.550		IOHS09
K.BETA DEC	00000.59341193	ORIENTATION ANGLE OF DELTA V VECTOR	
K0.650 DEC	00000.650		IOHSGB
K.40DG DEC	00000.6981317008	-ZP3 ~ L,A,O,R	
K000.9 DEC	00000.900	ZP8 ~ L,A	
K00001 DEC	00001		
K001.0 DEC	00001.0		
K.MUTE DEC	00001.0	DENSITY MUTILATION COEFFICIENT	N4DENS
K.INB2 DEC	00001.00673852	INVERSE OF MERC SPHEROID SEMIMINOR AXIS SQUARED	
K001.4 DEC	00001.4		
K00.A3 DEC	00001.41426667		
K.82DG DEC	00001.431169987		
K001.5 DEC	00001.50		
K1.525 DEC	00001.525		
KKKP12 DEC	00001.57079632675	HALF PI IN RADIANS	POSWIP
K01.75 DEC	00001.75		
K0.A5P DEC	00001.90782077		
K00002 DEC	00002		
K002.0 DEC	00002.0		
K0.A1P DEC	00002.28573082		
K002.5 DEC	00002.5		
KE.EEE DEC	00002.718281828		
K00003 DEC	00003		
K003.0 DEC	00003.0		
K00.PI DEC	00003.14159265		
K00004 DEC	00004		
K004.0 DEC	00004.		A0STAD
K03HPI DEC	00004.712388975	3 HALVES PI(RADIANS)	
K00005 DEC	00005		
K0005. DEC	00005.		
K005.0 DEC	00005.0		
KLPBSF DEC	00005.24615384		
K00006 DEC	00006		
K006.0 DEC	00006.0		
KKK2PI DEC	00006.283185307		
K0.A3P DEC	00006.55591931		

K00.UT DEC	00006.8104		
K00007 DEC	00007		
K00008 DEC	00008		
K00009 DEC	00009		
K00010 DEC	00010		
K010.0 DEC	00010		
K00011 DEC	00011		
K00012 DEC	00012		
K00013 DEC	00013		
KTHEMN DEC	00013.209693	THETA MEAN IN DEGREES	POSWIP
K013.4 DEC	00013.44684	MINUTES PER HG UNIT OF TIME	R5RARF
K00015 DEC	00015		
K00016 DEC	00016		
K00017 DEC	00017		
K00018 DEC	00018		
K00019 DEC	00019		
K00020 DEC	00020		
KS.UND DEC	00020.0463333	PROPORTIONALITY CONSTANT--CS-TM EQ	N4COEF
K00021 DEC	00021		
K00023 DEC	00023		IOMANI
K00024 DEC	00024		
K00025 PZE	00025		
K00026 DEC	00026		
K00027 DEC	00027		
K00028 DEC	00028		
K000JB PZE	00030		
K000NB PZE	00030		
K00030 DEC	00030		
K00031 DEC	00031		
K00032 DEC	00032		
K00033 DEC	00033		
K00034 DEC	00034		
K00035 DEC	00035		
K00037 DEC	00037		
K00039 DEC	00039		
K00040 DEC	00040		
K00041 DEC	00041		
K00042 DEC	00042		
K00045 DEC	00045		
K00046 DEC	00046		
K00047 DEC	00047		
K00048 DEC	00048		
K00051 DEC	00051		
K00052 DEC	00052		
K00053 DEC	00053		
K00054 DEC	00054		
K.1RAD DEC	00057.29577951	-L,A,O,R	
K00060 DEC	00060		
K0060. DEC	00060.		
K00063 DEC	00063		IOMANI
K00066 DEC	00066		
K00075 DEC	00075		
K00084 DEC	00084		MYSCRD
K00085 DEC	00085		
K00086 DEC	00086		
K00090 DEC	00090		
K00120 DEC	00120		
K0130. DEC	00130.	-ZP7 - L,A	
K0150. DEC	00150.		
K00154 DEC	00154		
K00155 DEC	00155		LAUNCH
K00156 DEC	00156		
K00160 DEC	00160		
KS.LNG DEC	00160.41646		

MC 102

K0.SF4 DEC	00162.8155068	SF4 - L,A,O,R	
K00163 DEC	00163		
K00164 DEC	00164		
K00165 DEC	00165		
K00166 DEC	00166		
K00167 DEC	00167		
K00168 DEC	00168		
K00179 DEC	00179		
K00180 DEC	00180		
K00198 DEC	00198		
K00218 DEC	00218		
K00219 DEC	00219		
K00220 DEC	00220		
K00221 DEC	00221		
K00222 DEC	00222		
K00224 DEC	00224		
K00256 DEC	00256		
KTFLMN DEC	00273.579971	TIME OF FALL MEAN IN SECONDS	POSWIP
KT.MPE DEC	00288.16	BASE MOLECULAR SCALE TEMP IN DEG K	N4COEF
K0300. DEC	00300.		
K0400. DEC	00400.	-ZP12-O,R	
K00420 DEC	00420		
K00512 DEC	00512	K10P - L,A,O,R	
K0600. DEC	00600.		
K0.SF6 DEC	00651.2620273	SF6 - L,A,O,R	
K0.SF3 DEC	00732.6697805	SF3 - L,A,O,R	
K806.8 DEC	00806.8134	SECONDS PER HG UNIT OF TIME	POSWIP
K01000 DEC	01000		
K01.E3 DEC	01000.0		
K01023 DEC	01023	KP3 - L,A,O,R	
K01024 DEC	01024	K12 - L,A	
K01800 DEC	01800		
KINTV2 DEC	02000	STORE EVERY 2000TH INTEGRATION STEP	R5RARF
K0.SF5 DEC	02442.232602	SF5 - L,A,O,R	
K2700. DEC	02700.		
K0.MPR DEC	03437.746771	-DGHR	
K0.SF1 DEC	05861.358244	SF1 - L,A	
KC.NVF DEC	06378.145	KM/HG UNITS OF LENGTH	N4DRAG
K07200 DEC	07200		
K.MACH DEC	07905.3827	HG UNITS OF LENGTH/HG UNITS OF TIME	N4COEF
K0.SF2 DEC	12787.5	SF2 - L,A	
K15360 DEC	15360	-DGHR	
KVELMN DEC	23306.4968	VELOCITY MEAN IN FT/SECOND	POSWIP
KCFTSC DEC	25936.294946	CONVERSION FACTOR TO FT/SEC	POSWIP
K32.E3 DEC	32000.0		
K0.9E4 DEC	90000.0		
K10.E5 DEC	100000.0		
K.N4HC DEC	136025.0	CRITICAL ALTITUDE DENSITY FORMULA	N4PDEN
KL.HGT DEC	141766.0		
KT.GHT DEC	172205.0		
K0.RAD DEC	6378145.	EQUATORIAL RADIUS OF EARTH IN METERS	N4DRAG
KMUYDS DEC	6975224.19	NUMBER OF YARDS IN ONE MERCURY UNIT	A0STAD
KCD.CV DEC	336939173.2		
KCD.CW DEC	381660578.3		

4.3.3 Octal Constants

K000P1 PON	0		
K000P4 FOR	0	MASK FOR A INDICATOR	LAUNCH
K000P6 SIX	0	MASK FOR A AND B INDICATORS	LAUNCH
K00377 OCT	377	ONES IN LAST 8 BITS	
KOMASK OCT	1	SENSE INDICATOR MASK	N4DENS

K10000 OCT	10000	MASK LIFTOFF BIT IN TELEMETRY	LAUNCH
K40003 OCT	40003	ENABLE DCC AND CHANNELS A + B	
KA05T1 DEC	.092,.090		
KA05T2 DEC	101.6,167.1		
KA7777 OCT	000000077777		
KAZ16M OCT	777776777777	MASK FOR OUTPUT BIT NUMBER 83	
KBB23Z OCT	000400000001	K8 - L,A,O,R	
KBB33Z OCT	200000000001	K4 - L,A	
KCBIT2 OCT	200000000000	MASK FOR LAUNCH PRINT SIGNAL (AB01)	
KCBIT4 OCT	040000000000	MASK FOR LAUNCH PRINT SIGNAL (CAPS)	
KCH200 OCT	200000000000		10HSGB
KCH201 OCT	201000000000		10HSGB
KCH211 OCT	211000000000		10HSGB
KCH232 OCT	232000000000	CONVERT FX TO FL PT + DIVIDE BY 2	
KCH233 OCT	233000000000		
KCH234 OCT	234000000000	-BCD	
KD0001 HTR	0,0,1		
KD0002 PZE	0,0,2		
KD0003 PZE	,,3		
KD0004 PZE	,,4	K5 - L,A,O,R	
KD0006 PZE	0,0,6		
KD0008 PZE	,,8		
KD0012 PZE	,,12	K13 - L,A	
KD0020 PZE	,,20	K1P - L,A	
KD0024 PZE	,,24	K2P - L,A,O,R	
KD0028 PZE	,,28	FPS-16 CODE FOR TMLANA+29	MPSTRP
KD0030 PZE	,,30	AZUZA CODE FOR TMLANA+29	MPSTRP
KD0040 PZE	,,40		POSWIP
KD0060 PZE	,,60	COLOR R (IR)	
KD1024 PZE	,,1024		
KD2048 PZE	,,2048	K9 - L,A	
KD7777 OCT	077777000000	K6 - L,A,O,R	
KDCMSK OCT	377777000000		
KDUMMY OCT	201400000000	MASK TO KEEP DECREMENT	POSWIP
KFPTBO PZE	20,0,48	DUMMY DELTA FOR FIRST ITERATION	
KH09RS OCT	1006	FIXED POINT TIME FOR BURNOUT	
KHGBRS OCT	1022	TIME CONVERSION CONSTANT	10HS09
KLSACM OCT	17777700	HS GEB INPUT TIME CONV CONSTANT	10HSGB
KMNMSK OCT	000777777777	MASK ACTIVATE LOW SPEED INPUT	
KMSK48 OCT	037000000000	- L,A	
KMTRRM OCT	34	MASK FOR BITS 4-8	
KPENUP OCT	74000000	TELEMETRY MASK RR BITS	
KRMSHS OCT	742	CONSTANT TO LIFT BOTH PENS	
KRVOHE PZE		HS INPUT TIME CONVERSION CONSTANT	10HSGB
KSBTST OCT	300000000	INDICATION OF RV OR HERGETS ELEMENTS	
KT0001 PZE	0,1,0	STAND BY TEST CONSTANT	
KT0004 PZE	,4		
KT0006 PZE	,6	K7P - L,A,O,R	
KTGMSK OCT	700000	K7 - L,A,O,R	
KU1777 OCT	233000001777	MASK TO KEEP TAG	
K000P5 FVE	0,0	K3 - L,A,O,R	
		LAUNCH	

4.4 TABLES

A machine-printout listing of tables is reproduced on the following pages. From left to right, the name of the table is listed first; the pseudo-operation and its variable field are presented next; third is an identification column, if applicable; last is printed the name of the program with which the table is associated.

MC 102

.....TABLES LAUNCH AND ORBIT.....

KLAMDO BSS	16,X		DODIFC
T000TA PZE			NOCJNI
T00IND PZE		TIME AT 450,000 FT(MIN/ADDR,SEC/DECR)	N4DRAG
TODCHC PZE		TIME AT 450,000 FT USED BY DC	N4DRAG
TODRAG PZE		ZERO=REENTRY, NONZERO FOR RETROFIRE	N4DRAG
T1STNO BSS	1,0	IF ZERO, 1ST ITERATION, OTHERWISE NOT	C9DTRF
T6DELT BSS	9,0	NOM TIME INTRVALS RETROS	R6BOTH
T6HOLD PZE	**,1,**		
TAWTPC BSS	88,F	WEIGHTED PARTIALS MAX NR LOC=K0000N*K000MB	
TCBEG1 BSS	7,0		
TCCONS BSS	220,,0	LAUNCH VARIABLE CONSTANTS	
TCCOUT BSS	30,0		
TCFP16 BSS	15,0		
TCIP71 BSS	15,0		
TCMANR BSS	250,X	TEMP STORAGE LAUNCHP	
TCORBP BSS	16,F		
TCR3GE BSS	8,0		
TCSUBD PZE		COEFFICIENT OF DRAG (HG UNITS)	N4DRAG
TD2SSM BSS	15,F	SUM(DF**2)/M, FOR EACH STATION	
TDCCNT BSS	24,0		DODIFC
TDEQCT BSS	24,F	THE PARTIALS OF R OR COP	DODIFC
TDFNDC PZE	180	DETERMINES K-1 OR K(0) TEST	
TDNIET PZE		LAST VALID TIME TO USE UINTP FOR INPT	DODIFC
TDNOBR BSS	10,X	NUMBER OF OBSERVATIONS PROCESSED	DODIFC
TDRANV SYN	KLAMDO+1	16 LOCNS IN TDRA5V, INPUT TO NOCPNI	
TDSUBX BSS	3,F	X COMP DRAG ACCELERATION (HG UNITS)	N4DRAG
TDSUBY SYN	TDSUBX+1	Y COMP DRAG ACCELERATION (HG UNITS)	
TDSUBZ SYN	TDSUBX+2	Z COMP DRAG ACCELERATION (HG UNITS)	
TDTIMT PZE		TIME TEST FOR END OF TABLE	DODIFC
TDTOBS PZE		TIME OB IN SEC FOR NDC	DODIFC
TEQCIN BSS	27,0		
TFESAB PZE		FIRST SECOND AFTER BURNOUT	
T.DLTA BSS	2,0	HG TIME BET BURNOUT AND NEXT WHOLE SECR5RARF	
TLSSSE BSS	17,F	SOLUT TO LEAST SQRS,SE,SE OF ELEMENTS D2SUNE	
TM8MNS BSS	19,0	MAXIMUM RADAR MSG TRANSMISSION TIMES	
TMALB1 BSS	MNNOBB*17,0	BUFFERS TABLES	MSLOGG
TMALB2 BSS	MNNOBB*17,0	BUFFER TABLES	MSLOGG
TMAREA BSS	14,0	LONGITS OF EMERG RECOV AREAS,FT PT RADR5RARF	
TMBBNI BSS	15,0	NUMERICAL INTEGRATION BUFFER BLOCK	MON
TMBFBK BSS	MNCHAR+10,0	BLOCK FOR READING ERR CORR STAT CHAR	MZSCRD
TMCHAR BSS	MNSCNO*MNCHAR+1,0	STAT. CHARAC. TABLE LENGTH+1 LOCN.	MON
TMCMDB BSS	14,S	TABLE OF I/O COMMANDS	MYMESS
TMCYCL BSS	M,S	CONTROLS TRANS TO CYCLED ROUTINES	MTHFSC
TMCYNO BSS	M,S	CONTAINS FREQ OF HANDLING ROUTINES	MTHFSC
TMDARE BSS	14,0	DELTA LONGIT CORRSP TO 5 MI IN REC A	R5RARF
TMDTBO DEC	22	DELTA TIME TO BURNOUT	
TMECT1 PZE			
TMECT2 PZE			
TMENDP BSS	2,0	LAT(+1=LONG)OF SUB-CAPS PT AT REENTRY	MON
TMERCN BSS	28,0		
TMERMC BSS	36,0		
TMETRP PZE			
TMETRS PZE			
TMFMSK BSS	12,0	MASKS FOR TURNING OFF INDICATORS	
TMFRPR BSS	N,S	CORRELATES PRIOR + ROUTS TABLE NRS	MOPRIO
TMGEB1 BSS	7,0		
TMGEDS BSS	2,0	DISCRETE AND TIME TAG	
TMGMT1 PZE			

TMGMT2 PZE			
TMH1DB BSS	7,0	GEB COMPUTED DATA OUTPUT BLOCK	LAUNCH
TMH2DB BSS	7,0	IP709 COMPUTED DATA OUTPUT BLOCK	LAUNCH
TMH1TM BSS	4,0	GEB DISCRETE SIGNAL OUTPUT BLOCK	LAUNCH
TMH2TM BSS	2,0	IP709 DISCRETE SIGNAL OUTPUT BLOCK	LAUNCH
TMHEDP BSS	7	TABLE HS RAW RADAR PARAMETERS	
TMHRAE BSS	301,0		
TMHS1T BSS	3,0	GEB TELEMETRY TIME OUTPUT BLOCK	LAUNCH
TMHS2T BSS	3,0	IP709 TELEMETRY TIME OUTPUT BLOCK	LAUNCH
TMHSL1 BSS	48,0	BUFFER FOR HS INPUT NR.1	MTHSI1
TMHSL2 BSS	48,0	BUFFER FOR HS INPUT NR.2	MTHSI2
TMIMPP BSS	10,0	IMPACT DATA FOR RETRO FIRING	N4DRAG
TMINRT BSS	8		
TMIP71 BSS	7,0		
TMLANA BSS	30,5	INPUT TABLE FOR OOLANA	OOLANA
TMLCD1 BSS	4,0		
TMLCD2 BSS	3,0		
TMLDLA BSS	4,0	DELTA LONGS REC AREAS	R5RARF
TMLMPT BSS	4,0	LONGS REC AREAS	R5RARF
TMLOUT BSS	4,0		
TMLSDB BSS	19,0	REFERA PARTIC STATIONS IN TMRM	IOTTIN
TMLSOX BSS	MNOLSY,0	FINAL ACQUIS. DATA OUTPUT BUFFER	OOLSTY
TMLSOY BSS	MNOLSY,0	FINAL ACQUIS. DATA OUTPUT BUFFER	OOLSTY
TMMES1 BSS	24,0	CARD IMAGE (MSGNR REQUESTED DISAGR W/1MYMSCK	
TMMES2 BSS	24,0	CARD IMAGE (MESSAGE REQUESTED NON-EXISMYMSCK	
TMMES3 BSS	24,0	BIN. INFO. CARD IMAGE BLOCK TO PRINT MYMSCK	
TMMESS BSS	25,0	LOC OF MESSAGE READ FROM TAPE	MYMSCK
TMMICL BSS	P,5	CONTROLS XFER TO CYCLED ROUT MYMINS	
TMMINO BSS	P,5	CONTAINS FREQ. OF HANDLING ROUT MYMINS	
TMMRLP BSS	2,0		
TMNMSK BSS	12,0	MASKS FOR TURNING ON INDICATORS	
TMNTRF BSS	4,0	TIMES TO FIRE FOR 1,2,3 ORBS	R5RARF
TMOGOD BSS	16,0	OUTPUT FROM ----- (PACKED WORD)	
TMOLAB BSS	13,0	OUTPUT FROM 00ORRE (PACKED DATA)	
TMORES BSS	500,0		MON
TMORMC BSS	36,0	OUTPUT TABLE	O5ORMC
TMORRE BSS	30,0	MAKES A DOUBLE BUFFER OUT OF TMORMC.	MPORRE
TMOX01 BSS	4		
TMOY01 BSS	4,0	DATA TABLE	MYTTOY
TMPANL BSS	11*N,0	MOSAVE OUTPUT, 11 LOCNS EA ROUTINE	MON
TMPRIO BSS	N,5	PRIORITY TABLE	MOPRIO
TMPRLG BSS	2,0	LOG BUFFER OF ON LINE OUTPUT	MON
TMQKEY BSS	N,5	QUEUEING TABLE, KEYS FOR VAR QUEUES	MOPRIO
TMQKY2 BSS	N,5	LIMITS NO OF QUEUE ENTS TO QUEUE	MOPRIO
TMRARF BSS	15,0		
TMREFR BSS	N,5	PRIORITY REFERENCE TABLE	MOPRIO
TMREST BSS	MNNWRB,0	INTERMEDIATE RESTART BUFFER	MYWRRS
TMRM18 BSS	203,0		
TMRM19 BSS	203,0		
TMRST1 BSS	MNNWRB+MNNWR1+MNNWR2,0	RESTART BUFFER	
TMRTCC BSS	31,0	TRANSFER TABLE IN ORDER BY SC NR	MORTCC
TMSAVE BSS	N,5	CONTINS 1ST LOC TO SAVE FOR EACH ROUT	MOPRIO
TMSLOP DEC	182	REL BET SPEED OF CAP CLOCK AND GMT	O5RARF
TMSSEC DEC	60180	PRESENT CAPSULE SETTING,SECONDS,B35	O5ORMC
TMSTAD BSS	16,0	GENER ACQUIS DATA(T,R,A,E)OOLSTY CONVMON	
TMSTCH BSS	37,0	REF TABLE FOR STAT CHAR	MYSCRD
TMSTMS BSS	15,0	REFERENCE TO TMRMES FOR DODIFC	MON
TMTFEA BSS	14,0	TIMES TO FIRE FOR EMERG RECOV AREAS	R5RARF
TMTML1 BSS	T,0	TELEMETRY AND TIME TAG LINE 1	
TMTML2 BSS	T,0	TELEMETRY AND TIME TAG LINE 2	
TMTTIN BSS	18,0	REF TABLE FOR TTY INPUT DATA BLOCKS	MTTIN
TN4LLH BSS	3,5	OUTPUT BLOCK FOR A2CSCP	N4DRAG
TN4NEX PON		SETONE CHANGED ONLY DURING REENTRY	N4DRAG
TN7ICT PZE		NO OF TIMES CONV TEST FAILED IN N2EXCR	

MC 102

TN7N12 PZE		INDICATOR,0=FROM N1SOFT,1=N2EXCR	N7VARS
TNDCIN PZE		INDICATOR FOR METHOD TO CALC PARTIALS	DODIFC
TNDRAG BSS	7,0	INPUT BLOCK FOR N4DRAG	N4DRAG
TNEARM PZE		ANCHOR POINT FIXED POINT SECONDS	N0CPNI
TNETIM PZE		TIME OF LAST ENTRY IN N1 TABLE	N0CPNI
TNFUNK BSS	40,F	INTERMEDIATE OUTPUT TABLE	N0CPNI
TNIINT BSS	9,0	INTERMEDIATE OUTPUT TABLE	N0CPNI
TNINT1 BSS	2200,0	INTEGRATION (REGULAR) TABLE	N0CPNI
TNINT2 BSS	1800,0	INTEGRATION (ABORT) TABLE	
TNINT3 BSS	112,0	FIRST LOCATION PERTURBED X TABLE	
TNINT4 BSS	112,0	FIRST LOCATION PERTURBED Y TABLE	
TNINT5 BSS	112,0	FIRST LOCATION PERTURBED Z TABLE	R1NDPC
TNINT6 BSS	112,0	FIRST LOCATION PERTURBED X DOT TABLE	R1NDPC
TNINT7 BSS	112,0	FIRST LOCATION PERTURBED Y DOT TABLE	R1NDPC
TNINT8 BSS	112,0	FIRST LOCATION PERTURBED Z DOT TABLE	R1NDPC
TNINT9 BSS	10,0	REENTRY TABLE FOR TIME TO FIRE CALC.	R5RARF
TNNDIC BSS	15,0	15 WORD INPUT BLOCK TO NOCPNI	R1NDPC
TNOMFS BSS	15,0	NO. OF OBSERVATIONS DESIRED FROM EACH STATION	
TNSAVR BSS	6	LOC. OF LAST VALID RV	N7VARS
TNSCOR BSS	17,X		
TNUMNI PZE		N0CPNI TESTS FOR TABLE NR (TNINT-)	R1NDPC
TNVS.C BSS	18,0	KS FOR COMPUTING F(1) FOR HALF-STEP	N7VARS
TODCHC PZE		TIME AT 450,000 FT.	
TOMEGD PZE		VALUE OF OMEGA DOT	DODIFC
TPA.RX BSS	6,F		
TPWTFD BSS	11,F	W(DF)MAX.NR. OF LOCNS=*(K000MB)	
TRAVAD BSS	17,0		
TRERTH DEC	1,0	RADIUS OF THE EARTH	POSWIP
TRVIBO BSS	7,0	OUTPUT FROM U7INTP TO R6BOTH	R5RARF
TRVTAP BSS	13,F		DODIFC
TSETCT BSS	12,0		DODIFC
TSTWTS BSS	33,0	WEIGHT WORD TABLE	DODIFC
TSVSEE BSS	6,F		
TTHRUS BSS	9,0		R6BOTH
TTIMES BSS	9,0		R6BOTH
TVELOC PZE		VELOCITY VECTOR MAGNITUDE (HG UNITS)	N4DRAG
TZZZZZ BSS	0		
ORG		KLAMDO	
DEC	1.406272864		
ORG	T6DELT	BSS 9	
DEC	12.,5.,7.,5.,5.,5.,2.,5.		
ORG	TCCONS		
DEC	.5	DELTA T SUB ND	0
DEC	2.5	F(SECONDS)	1
DEC	2.0	G(SECONDS)	2
DEC	0.0	H(SECONDS)	3
DEC	.9557638	CONVERSION GE-B TO HG(R)	4
DEC	1.022883455	CONVERSION GE-B TO HG(V)	5
DEC	1.00003643		6
DEC	1.000009664	CONVERSION IP-709 TO HG(V)	7
DEC	1.0463218	CONVERSION IP-709 TO GE-B(R)	8
DEC	.97763793004	CONVERSION IP-709 TO GE-B(V)	9
DEC	266.0	C(SECONDS)	10
DEC	-.2620152644	B RADIANS/GE-B(V)	11
DEC	0	K4 89 FOR	
DEC	0	K3 89 S NOM	
DEC	0	K2 89	
DEC	0	K1 89	
DEC	0	K0 89	
DEC	3.651613E-10	K5 84 (V/VR)NOM	17
DEC	0.0	K4 84 BELOW	18
DEC	1.308742E-05	K3 84 STAGING	19
DEC	-1.532237E-04	K2 84	20
DEC	0.05059132	K1 84	21

DEC	4.844717E-13	K13 84 (V/VR)NOM	22
DEC	-0.2195182E-9	K12 84 ABOVE	23
DEC	1.614467E-8	K11 84 STAGING	24
DEC	1.219781E-5	K10 84	25
DEC	-0.001598267	K9 84	26
DEC	0.3885452	K8 84	27
DEC	1.0455	RADIUS OF EARTH GE-B	28
DEC	3291.58325	CONVERSION GE-B TO N. MILES	29
DEC	250.0	TIME TO FIRE RETROS & HIT AREA A	30
DEC	2.5	TRANSMISSION DELAT TIME	31
DEC	.0095576568	A(200,000 FT)HG.	32
DEC	.058833543	OMEGA RAD/HG UNIT OF TIME	33
DEC	.2389409468E-3	HF1 (5000 FT - MERCURY UNITS)	34
DEC	.005400076092	HF2(113,000 FT)HG	35
DEC	.999251039	RADIUS OF EARTH HG	36
DEC	3.5	L(SECONDS)	37
DEC	20925672.5	CONVERT HG TO FEET	38
DEC	-1.5819827	K1 81	39
DEC	152764502.9	K2 81	40
DEC	.729211508E-4	OMEGA E RAD/SEC	41
DEC	5.054415	LONG. OF PAD AT LIFTOFF	42
DEC	.493652887	RADIANS - GEOCENTRIC LAT. OF RADAR3	43
DEC	.999251039	(HG)DISTANCE FROM GEOCENTER TO PAD(RO)	44
DEC	-1.406416522	LP PRIME(80.581731) (RADIANS)	45
DEC	1.57079632679	PI/2 RADIANS	46
DEC	1.25957209	BETA ZERO	47
DEC	3443.929	CONVERSION HG TO N. MILES	48
DEC	26529.807	CONVERSION GE-B TO FT/SEC	49
DEC	4096	MASK	50
DEC	2048	MASK	51
DEC	1024	MASK	52
DEC	512	MASK	53
DEC	256	MASK	54
DEC	128	MASK	55
DEC	64	MASK	56
DEC	32	MASK	57
DEC	16	MASK	58
DEC	8	MASK	59
DEC	4	MASK	60
DEC	2	MASK	61
DEC	1	MASK	62
DEC	-.32619444E-04	OEFFICIENTS TO CURVE FIT	63
DEC	+.82729604E-03	ELOCITY OF ESCAPE ROCKET	64
DEC	-.53157966E-02	S A FUNCTION OF ALTITUDE,	65
DEC	+.19077357E-01	.E., BELOW 80,000 FT(HG)	66
DEC	-.9845195E-05	COEFFICIENTS TO CURVE FIT	67
DEC	+.58733290E-03	ELOCITY OF ESCAPE ROCKET	68
DEC	+.12643158E-01	BOVE 80,000 FT(HG)	69
DEC	+.00382306272	80,000 FT(HG)	70
DEC	25936.294946	CONVERSION HG TO FT/SEC	71
DEC	-.032338615	COEFFICIENTS TO OBTAIN	72
DEC	+.04744618	H MIN RET AS A FUNCTION	73
DEC	+.006608992	OF VELOCITY(V12 MAG)HG	74
DEC	30.0	30 SECONDS	75
DEC	.912235153	B (VELOCITY 23,660) MERCURY UNITS	76
DEC	.02150472780	450,000 FT(HG)	77
DEC	-5.071115E-13	K21 83	78
DEC	-5.427866E-13	K20 83 GAMMA	79
DEC	4.209581E-8	K19 83 NOM	80
DEC	-6.668477E-6	K18 83 BELOW	81
DEC	2.652569E-4	K17 83 STAGING	82
DEC	0.007767488	K16 83	83
DEC	0.1331041E-9	K26 83 GAMMA	84
DEC	-0.1168537E-6	K25 83 NOM	85

MC 102

DEC	0.3807413E-4	K24 83 ABOVE	86
DEC	-0.007518138	K23 83 STAGING	87
DEC	0.9371001	K22 83	88
DEC	0.0	ALPHA(SECONDS)	89
DEC	.732564155E-3	VP (MERCURY UNITS)	90
DEC	14.747978	K3 50 COEFFICIENTS TO	91
DEC	11.267406	K2 50 OBTAIN ACCEL. VS	92
DEC	3.7876281	K1 50 VELOCITY(GE-B)	93
DEC	.00476474885	DELTA LONG(.273 TOLERANCE)RAD	94
DEC	900.0	ARTIFICIAL TIME TO FIRE RETRO ROCKETS	95
DEC	130.0	S STRIP SECONDS	96
DEC	3441.35029	RADIUS OF EARTH N. MILES	97
DEC	23391.3384	MEAN VALUE OF VEL. FT/SEC	98
DEC	-4.1594	MEAN VALUE OF GAMMA	99
DEC	.28548238E-9	EMPIRICAL	100
DEC	.68265288E-9	CURVE	101
DEC	.62475098E-6	FIT	102
DEC	.59100400E-5	CONSTANTS	103
DEC	.31345620E-5		104
DEC	-.43086694E-2	EMPIRICAL	105
DEC	.23747061E-1	CURVE	106
DEC	.44627038E-1	FIT	107
DEC	.66648080	CONSTANTS	108
DEC	.27185713E+1		109
DEC	.14873996E+2		110
DEC	.86767697E+2		111
DEC	581.242093		112
OCT	001000000000	1 IN CHARACTERISTIC	113
DEC	0.7853981634		114
DEC	0.1396263401	MAX. PLOT BOARD VALUE RAD.	115
DEC	0	START OF TABLE	116
DEC	5.50651379	LONG OF AREA A RAD.	117
DEC	5.56760031	LONG OF AREA B RAD.	118
DEC	5.65312144	LONG OF AREA C RAD.	119
DEC	5.77878515	LONG OF AREA D RAD.	120
DEC	0	LONG OF AREA E	121
DEC	7.033676885	LONG OF AREA XB RAD.	122
DEC	50.	END OF TABLE	123
DEC	0	NOT USED IN LAUNCH	124
DEC	0	NOT USED IN LAUNCH	125
DEC	0	NOT USED IN LAUNCH	126
DEC	6.00393262	LONG OF AREA E RAD.	127
DEC	.334517988E-3	EPSILON(7,000)HG	128
DEC	.570722665	(32.7)INCLINATION ANGLE DEVIATION(R)	129
DEC	806.8104	HG TO SECONDS	130
DEC	0	CODE FOR AREA A	131
DEC	1	CODE FOR AREA B	132
DEC	2	CODE FOR AREA C	133
DEC	3	CODE FOR AREA D	134
DEC	4	CODE FOR AREA E	135
DEC	9	CODE FOR AREA XB	136
DEC	0	NOTUSED IN LAUNCH	137
DEC	0	NOTUSED IN LAUNCH	138
DEC	0	NOTUSED IN LAUNCH	139
DEC	0	NOTUSED IN LAUNCH	140
DEC	0	NOTUSED IN LAUNCH	141
DEC	0	NOTUSED IN LAUNCH	142
DEC	5.93411945	LONG AREA E PRIME(RAD)	143
OCT	000074000000	60 AT BINARY SCALE OF 17	144
DEC	2.0	DELTA T USED IN V/VR CALCULATIONS	145
DEC	753.869034	CONVERT SEC TO GE-B UNITS	146
DEC	-1.40581257	LONGITUDE OF PAD (-80.547114 DEGREES)	147
DEC	.497259538	GEODETIC LAT. OF PAD (28.4908729)	148
DEC	20000000.	CONVERT FROM GE-B TO FEET	149

DEC	0		
DEC	0		
DEC	0		
DEC	0		
DEC	-0.00044	ORBIT LIFETIME CONSTANTS K1,7	154
DEC	+0.00042	ORBIT LIFETIME CONSTANTS K2,7	155
DEC	+0.00625	ORBIT LIFETIME CONSTANTS K3,7	156
DEC	-0.00020	ORBIT LIFETIME CONSTANTS K4,7	157
DEC	+0.998142	ORBIT LIFETIME CONSTANTS K5,7	158
DEC	-0.00047	ORBIT LIFETIME CONSTANTS K1,6	159
DEC	+0.00013	ORBIT LIFETIME CONSTANTS K2,6	160
DEC	+0.00623	ORBIT LIFETIME CONSTANTS K3,6	161
DEC	-0.00013	ORBIT LIFETIME CONSTANTS K4,6	162
DEC	+0.998028	ORBIT LIFETIME CONSTANTS K5,6	163
DEC	-0.00010	ORBIT LIFETIME CONSTANTS K1,5	164
DEC	+0.00025	ORBIT LIFETIME CONSTANTS K2,5	165
DEC	+0.00595	ORBIT LIFETIME CONSTANTS K3,5	166
DEC	-0.00015	ORBIT LIFETIME CONSTANTS K4,5	167
DEC	+0.997925	ORBIT LIFETIME CONSTANTS K5,5	168
DEC	-0.00005	ORBIT LIFETIME CONSTANTS K1,4	169
DEC	+0.00023	ORBIT LIFETIME CONSTANTS K2,4	170
DEC	+0.00578	ORBIT LIFETIME CONSTANTS K3,4	171
DEC	-0.00014	ORBIT LIFETIME CONSTANTS K4,4	172
DEC	+0.997802	ORBIT LIFETIME CONSTANTS K5,4	173
DEC	-0.00007	ORBIT LIFETIME CONSTANTS K1,3	174
DEC	+0.00016	ORBIT LIFETIME CONSTANTS K2,3	175
DEC	+0.00571	ORBIT LIFETIME CONSTANTS K3,3	176
DEC	-0.00012	ORBIT LIFETIME CONSTANTS K4,3	177
DEC	+0.997665	ORBIT LIFETIME CONSTANTS K5,3	178
DEC	-0.00016	ORBIT LIFETIME CONSTANTS K1,2	179
DEC	+0.00005	ORBIT LIFETIME CONSTANTS K2,2	180
DEC	+0.00565	ORBIT LIFETIME CONSTANTS K3,2	181
DEC	-0.00007	ORBIT LIFETIME CONSTANTS K4,2	182
DEC	+0.997493	ORBIT LIFETIME CONSTANTS K5,2	183
OCT	30000	3 POS1. ROC. FIRED PRINT SIGNAL	184
OCT	27000	2 POS1. ROC. FIRED PRINT SIGNAL	185
OCT	26000	1 POS1. ROC. FIRED PRINT SIGNAL	186
OCT	25000	0 POS1. ROC. FIRED PRINT SIGNAL	187
OCT	23000	NO GO IS RECOMMENDED PRINT SIGNAL	188
OCT	22000	GO IS RECOMMENDED PRINT SIGNAL	189
OCT	21000000	10 PNTS USED TO CALC. FINAL GO-NO-GO	190
OCT	20000000	9 PNTS USED TO CALC. FINAL GO-NO-GO	191
OCT	17000000	8 PNTS USED TO CALC. FINAL GO-NO-GO	192
OCT	16000000	7 PNTS USED TO CALC. FINAL GO-NO-GO	193
OCT	15000000	6 PNTS USED TO CALC. FINAL GO-NO-GO	194
OCT	14000000	5 PNTS USED TO CALC. FINAL GO-NO-GO	195
OCT	13000000	4 PNTS USED TO CALC. FINAL GO-NO-GO	196
OCT	12000000	3 PNTS USED TO CALC. FINAL GO-NO-GO	197
OCT	11000000	2 PNTS USED TO CALC. FINAL GO-NO-GO	198
OCT	10000000	1 PNTS USED TO CALC. FINAL GO-NO-GO	199
OCT	7000000	0 PNTS USED TO CALC. FINAL GO-NO-GO	200
OCT	24000	INSUFFICIENT DATA TO MAKE GO-NOGO REC.	201
DEC	0.84	BREAK POINT FOR TAIL OFF ACCERL FIT	202
DEC	50.872771	COEFFICIENTS FOR	203
DEC	-83.680596	TAIL OFF ACCELERATION	204
DEC	34.800786	FIRST HALF OF CURVE	205
DEC	.029344515	2ND HALF OF CURVE	206
DEC	-.25351989		207
DEC	.55917158		208
DEC	1.0068147688		209
DEC	-.113238E-3	CONSTANTS FOR VGO CALC	210
DEC	.320393E-4		211
DEC	.551194E-2		212
DEC	-.687252E-4		213

```

DEC      0.9977404                                214
PZE      2399,0,59                                215
ORG      TCCOUT+5
DEC      .497418836                                LAT CAPE PAD COORD
DEC      -1.404990047                              LONG CAPE PAD COORD
DEC      .497418836                                LAT CAPE PAD COORD
DEC      -1.404990047                              LONG CAPE PAD COORD
ORG      TCFP16+13
PZE      TCCOUT+1
PZE      TCCOUT+3
ORG      T.DLTA                                BSS 2
DEC      0,0
ORG      TCIP71+13
PZE      TCCOUT+1
PZE      TCCOUT+3
ORG      TCR3GE+6
PZE      TCCOUT
PZE      TCCOUT+2
ORG      TMAREA                                BSS 14
DEC      0.7504916,.1.9722221,3.5430184,.4.2935100,5.9690260
DEC      0.7155850,.1.9722221,3.5255651,.4.2236968,5.9515727
DEC      0.6457718,.1.9373155,3.5255651,.4.2062435
ORG      TMCMDA                                BSS 14
IOCPN    0,0,**                                POSITION TAPE
IOCT      TMMES5,0,25                          READ 25 WORD MSG FROM TAPE
IOCP      TMMES2,0,24                          MSG NO REQUESTED OUT OF RANGE
IOCP      TMMES3,0,24                          NO REQUESTED IN BINARY
IORP      0,0,0
IORP      0,0,0
IORP      0,0,0
IORT      0,0,0
IOCP      TMMES1,0,24                          MSG NO FOUND AND REQSTED DISAGREE
IOCP      TMMES3,0,24                          NO REQUESTED IN BINARY
IOCP      TMMES5+1,0,24                        PRINT MSG READ FROM TAPE
TCH      TMCMDA+4
IOCP      TMMES1,0,24                          MSG NO FOUND AND REQSTED DISAGREE
TCH      TMCMDA+4
ORG      TMCYCL                                BSS M,S
PZE      MNORMC,0,24
PZE      MNMINS,0,120                          MUST BE LAST ENTRY IN TMCYCL
ORG      TMCYNO                                BSS M,S
PZE      0,0,24
PZE      0,0,120                              MUST BE LAST ENTRY IN TMCYNO
ORG      TMDARE                                BSS 14
DEC      .000890,.001047,.000873,.000960,.000925,.000925,.001030
DEC      .000908,.001013,.000925,.000978,.000978,.000978,.001047
ORG      TMERMC                                BSS 36
DEC      0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0
DEC      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
ORG      TMFMSK                                BSS 12  TRNOF MASKS
OCT      377777777777                          TRNOF A
OCT      577777777777                          TRNOF B
OCT      677777777777                          TRNOF C
OCT      737777777777                          TRNOF D
OCT      757777777777                          TRNOF E
OCT      767777777777                          TRNOF F
OCT      773777777777                          TRNOF G
OCT      775777777777                          TRNOF H
OCT      776777777777                          TRNOF J
OCT      777377777777                          TRNOF K
OCT      600377777777                          TRNOF  C THRU K
OCT      177777777777                          TRNOF  A AND B
ORG      TMHEDP
PZE      TMHRAE+1

```

PZE	TMHRAE+1		2
PZF	TMHRAE+1		3
PZE	TMHRAE+2		4
PZE	TMHRAE+2		5
PZE	TMHRAE+3		6
PZE	TMHRAE+3		7
ORG	TMINRT		
PZE	TNINT2,0,0	BLOCK TO INITIALIZE TMRARF	1
DEC	8	FOR LAUNCH REENTRY	2
DEC	1		3
DEC	24		4
DEC	2368		5
DEC	0		6
DEC	1		7
DEC	1		8
ORG	TMLANA+5		
DEC	.497418836	LAT CAPE PAD COORD	
DEC	-1.404990047	LONG CAPE PAD COORD	
DEC	.497418836	LAT CAPE PAD COORD	
DEC	-1.404990047	LONG CAPE PAD COORD.	
ORG	TMLCD1	BSS 4	
PZE	,,12	IP709 CODE FOR TCCOUT+4	
PZE	,,12	IP709 CODE FOR TCCOUT+4	
PZE	,,8	RAW FPS-16 CODE FOR TCCOUT+4	
PZE	,,4	GE CODE FOR TCCOUT+4	
ORG	TMLCD2	BSS 3	
PZE	,,24	IP709 (FPS-16) FOR TCCOUT+29	
PZE	,,28	IP709 (AZUZA) FOR TCCOUT+29	
PZE	,,24	RAW FPS-16 CODE FOR TCCOUT+29	
ORG	TMLDLA	BSS 4	
DEC	.001030,.000978,.000943,.000943		
ORG	TMLMPT	BSS 4	
DEC	5.3302356,5.2202798,5.1050881,5.1050881		
ORG	TMLOUT	BSS 4	
PZE	CCSTIP	IP709 ENTRANCE FOR STRIP CHARTS	
PZE	CCSTIP	IP709 ENTRANCE FOR STRIP CHARTS	
PZE	CCST16	RAW ENTRANCE FOR STRIP CHARTS	
PZE	CCSTGE	GE ENTRANCE FOR STRIP CHARTS	
ORG	TMLSDB	BSS 19	
PZE	TMRM01		
PZE	TMRM02		
PZE	TMRM03		
PZE	TMRM04		
PZE	TMRM05		
PZE	TMRM06		
PZE	TMRM07		
PZE	TMRM08		
PZE	TMRM09		
PZE	TMRM10		
PZE	TMRM11		
PZE	TMRM12		
PZE	TMRM13		
PZE	TMRM14		
PZE	TMRM15		
PZE	TMRM16		
PZE	TMRM17		
PZE	TMRM18		
PZE	TMRM19		
ORG	TMMES1	BSS 24	
VFD	036/100021010,036/600000		
VFD	036/20000002,036/204000000		
VFD	036/10000002000,036/1000000		
VFD	036/400000020,036/430020000300		
VFD	036/205044400600,036/302000121000		

MC 102

```

VFD      036/400010240000,036/5000010000
VFD      036/1000004,036/210040400
VFD      036/140002010100,036/40002000
VFD      036/20000004000,036/0
VFD      036/140013010124,036/4210040100
VFD      036/401520001000,036/612021500600
VFD      036/234044666612,036/121044233000
ORG      TMMES2          BSS 24
VFD      036/220000300000,036/400000002000
VFD      036/10000000000,036/100000000000
VFD      036/40400000,036/4000
VFD      036/2000150,036/4100000
VFD      036/101024050400,036/222002210000
VFD      036/2000004000,036/41110000000
VFD      036/20200,036/4040040400
VFD      036/400600001002,036/10201001000
VFD      036/40100000004,036/0
VFD      036/10600020052,036/54141045400
VFD      036/122006640300,036/500004300000
VFD      036/641160115404,036/223212012000
ORG      TMMICL          BSS P
PZE      MNWRRS,0,MNDNMN
ORG      TMMINO          BSS P
PZE      0,0,MNDNMN
ORG      TMNMSK          BSS 12  TRNON MASKS
OCT      400000000000    TRNON A
OCT      200000000000    TRNON B
OCT      100000000000    TRNON C
OCT      400000000000    TRNON D
OCT      200000000000    TRNON E
OCT      100000000000    TRNON F
OCT      400000000000    TRNON G
OCT      200000000000    TRNON H
OCT      100000000000    TRNON J
OCT      400000000000    TRNON K
OCT      177400000000    TRNON C THRU K
OCT      600000000000    TRNON A AND B
ORG      TMNTRF          BSS 4
DEC      49680,55080,60480,60480
ORG      TMOX01
PZE      0,0,4
PZE      0,0,0
PZE      0,0,0
PZE      TMLS0X,,31
ORG      TMOY01          BSS 4
PZE      **              STATION NUMBER EST HORIZON GROSS TIME
PZE      **              S.C. MASK FROM STATION CHARACTERISTICS
PZE      **              LOGGING TIME FROM AJACQ MACRO
PZE      TMLSOY,0,**0    WORD COUNT LOC OUTPUT BLOCK OF OOLSTY.
ORG      TMRARF          BSS 15
DEC      0,0,0,0,0,0,0,0,8,1,0,2368,0,1,0
ORG      TMRTCC          BSS 31
PZE      MTSENS,0,TMSENS SENSE OUTPUT-31
PZE      MTTTIN,0,TMTI17
PZE      MTTTIN,0,TMTI16  TTY IN 16 S.C. 29
PZE      MTTTIN,0,TMTI15  TTY IN 15 S.C. 28
PZE      MTTTIN,0,TMTI14  TTY IN 14 S.C. 27
PZE      MTTTIN,0,TMTI13  TTY IN 13 S.C. 26
PZE      MTTTIN,0,TMTI12  TTY IN 12 S.C. 25
PZE      MTTTIN,0,TMTI11  TTY IN 11 S.C. 24
PZE      MTTTIN,0,TMTI10  TTY IN 10 S.C. 23
PZE      MTTTIN,0,TMTI09  TTY IN 9 S.C. 22
PZE      MTTTIN,0,TMTI08  TTY IN 8 S.C. 21
PZE      MTTTIN,0,TMTI07  TTY IN 7 S.C. 20

```

2
3
4

```

PZE      MTTTIN,0,TMTI06      TTY IN  6 S.C. 19
PZE      MTTTIN,0,TMTI05      TTY IN  5 S.C. 18
PZE      MTTTIN,0,TMTI04      TTY IN  4 S.C. 17
PZE      MTTTIN,0,TMTI03      TTY IN  3 S.C. 14
PZE      MTTTIN,0,TMTI02      TTY IN  2 S.C. 15
PZE      MTTTIN,0,TMTI01      TTY IN  1 S.C. 14
PZE      OPEN                  S.C. 13
PZE      OPEN                  S.C. 12
PZE      MTTTOY,0,TMYBOX      TTY OUT 2 S.C. 11
PZE      MTTTOX,0,TMXBOX      TTY OUT 1 S.C. 10
PZE      MTINTV,0,TMINTV      INTERVAL TIMER S.C. 09
PZE      MTWVVI,0,0           WWV TRAP S.C. 08
PZE      MTHFSC,0,TM8.3M      1/2 SECOND TRAP S.C. 07
PZE      OPEN                  6
PZE      OPEN                  5
PZE      MTHSOP,C,TMHSOP      PLOTTER - 4
PZE      MTHSOD,0,TMHSOD      DISPLAYS - 3
PZE      MTHS09,0,TMHS09      HS IN 2-2
PZE      MTHSGB,0,TMHSGB      MHS IN 1-1
ORG      TMSTCH               BSS 37
OCT      777777777777       WILL BE ZERO WHEN ST. CH. BLOCK IS LOA
PZE      TMCHAR-1
PZE      TMCHAR+MNCHAR-1
PZE      TMCHAR+2*MNCHAR-1
PZE      TMCHAR+3*MNCHAR-1
PZE      TMCHAR+4*MNCHAR-1
PZE      TMCHAR+5*MNCHAR-1
PZE      TMCHAR+6*MNCHAR-1
PZE      TMCHAR+7*MNCHAR-1
PZE      TMCHAR+8*MNCHAR-1
PZE      TMCHAR+9*MNCHAR-1
PZE      TMCHAR+10*MNCHAR-1
PZE      TMCHAR+11*MNCHAR-1
PZE      TMCHAR+12*MNCHAR-1
PZE      TMCHAR+13*MNCHAR-1
PZE      TMCHAR+14*MNCHAR-1
PZE      TMCHAR+15*MNCHAR-1
PZE      TMCHAR+16*MNCHAR-1
PZE      TMCHAR+17*MNCHAR-1
PZE      TMCHAR+18*MNCHAR-1
PZE      TMCHAR+19*MNCHAR-1
PZE      TMCHAR+20*MNCHAR-1
PZE      TMCHAR+21*MNCHAR-1
PZE      TMCHAR+22*MNCHAR-1
PZE      TMCHAR+23*MNCHAR-1
PZE      TMCHAR+24*MNCHAR-1
PZE      TMCHAR+25*MNCHAR-1
PZE      TMCHAR+26*MNCHAR-1
PZE      TMCHAR+27*MNCHAR-1
PZE      TMCHAR+28*MNCHAR-1
PZE      TMCHAR+29*MNCHAR-1
PZE      TMCHAR+30*MNCHAR-1
PZE      TMCHAR+31*MNCHAR-1
PZE      TMCHAR+32*MNCHAR-1
PZE      TMCHAR+33*MNCHAR-1
PZE      TMCHAR+34*MNCHAR-1
PZE      TMCHAR+35*MNCHAR-1
ORG      TMSTMS               BSS 15
MZE      0
MZE      0
MZE      0
MZE      0
MZE      0
MZE      0

```

MC 102

MZE	0		
MZE	0		
MZE	0		
MZE	0		
MZE	0		
MZE	0		
MZE	0		
MZE	0		
MZE	0		
ORG	TMTFEA	BSS 14	
DEC	45390,46470,47880,48618,49992,51060,52122		
DEC	53568,54180,55650,56640,57660,59160,59760		
ORG	TMTTIN	BSS 18	
PZE			
MZE	AAS01,1,10		
MZE	AAS02,1,10		
MZE	AAS03,1,10		
MZE	AAS04,1,10		
MZE	AAS05,1,10		
MZE	AAS06,1,10		
MZE	AAS07,1,10		
MZE	AAS08,1,10		
MZE	AAS09,1,10		
MZE	AAS10,1,10		
MZE	AAS11,1,10		
MZE	AAS12,1,10		
MZE	AAS13,1,10		
MZE	AAS14,1,10		
MZE	AAS15,1,10		
MZE	AAS16,1,10		
MZE	AAS17,1,6		
ORG	TNSCOR	BSS 17	
DEC	1,1,1,2,2,3,4,5,6,6,7,7,8,9,10,11,11		
ORG	TNVS.C		
DEC	.24609375	1	
DEC	-.02734375	2	
DEC	.01171875	3	
DEC	1.23046875	4	
DEC	.41015625	5	
DEC	-.09765625	6	
DEC	-.8203125	7	
DEC	.8203125	8	
DEC	.5859375	9	
DEC	.4921875	10	
DEC	-.2734375	11	
DEC	.5859375	12	
DEC	-.17578125	13	
DEC	.08203125	14	
DEC	-.09765625	15	
DEC	.02734375	16	
DEC	-.01171875	17	
DEC	.01171875	18	
ORG	TSTWTS	BSS 33	
VFD	4/7,4/3,4/2,4/2,12/2	PASS 1	CANAVERAL 000
PZE			BERMUDA 001
VFD	4/15		CANARY 002
VFD	4/8,4/8		MUCHEA 003
VFD	4/6,4/5,4/5		WOOMERA 004
VFD	4/6,4/3,4/3,4/3		*HAWAII 005
VFD	8/8,4/8		*PT. ARGUELLO 006
VFD	12/8,4/8		GUAYMAS 007
VFD	4/6,12/6,4/4		WHITE SANDS 008
VFD	4/6,4/5,12/3,4/2		
VFD	4/6,4/6,4/2,12/2		EGLIN 010

VFD	4/6,4/2,4/4,4/1,4/1,4/1,8/1	PASS2 CANAVERAL	011
VFD	4/4,4/5,4/3,4/3,4/1	BERMUDA	012
VFD	4/8,4/2,4/3,4/1,4/1,4/1	CANARY	013
VFD	4/4,4/5,4/3,4/4	MUCHEA	014
VFD	4/8,4/2,4/3,4/1,4/2	WOOMERA	015
VFD	4/4,4/7,4/2,4/3	HAWAII	016
VFD	4/3,4/3,4/7,4/3	PT. ARGUELLO	017
VFD	4/4,4/3,4/3,4/4,4/2	GUAYMAS	018
VFD	4/4,4/4,4/2,4/1,4/4	WHITE SANDS	019
VFD	4/6,4/3,4/3,4/1,4/1,4/2	CORPUS CHRISTI	020
VFD	4/4,4/7,4/1,4/1,4/1,4/1,4/1	EGLIN	021
VFD	4/8,4/1,4/4,4/1,4/1,4/1	PASS 3 CANAVERAL	022
VFD	4/4,4/5,4/1,4/3,4/1,4/1,4/1	BERMUDA	023
VFD	4/4,4/5,4/1,4/3,4/1,4/1,4/1	*CANARY	024
VFD	8/4,4/4,4/4,8/2,4/1,4/1	MUCHEA	025
VFD	4/4,8/4,4/4,4/2,8/1,4/1	*WOOMERA	026
VFD	8/8,8/8	HAWAII	027
VFD	4/8,8/8	PT. ARGUELLO	028
VFD	4/8,4/5,8/3	GUAYMAS	029
VFD	4/3,4/7,4/5,8/1	WHITE SANDS	030
VFD	4/8,4/2,4/3,4/2,8/1	CORPUS CHRISTI	031
VFD	4/4,4/8,4/1,4/2,4/1	EGLIN	032
ORG	TTHRUS	BSS 9	
DEC	1.,1.,2.,1.,1.,2.,3.,2.,1.	THRUST FACTOR TABLE	R6BOTH
ORG	TTIMES	BSS 9	
DEC	.014873383.,.006197243.,.0086761402	MERC UNITS TIME	R6BOTH
DEC	.006197243.,.006197243.,.006197243	THRST TIMING	R6BOTH
DEC	.0024788972.,.006197243.,.006197243	TABLE	R6BOTH
ORG	TZZZZZ		

4.5 COMMUNICATION CELLS

Represented below is an actual machine listing of communication cells—core storage cells used to communicate information from one self-contained routine to another self-contained routine.

The name of the communication cell is listed in the first column; in some cases a pseudo-operation (PZE) is indicated immediately following the cell name. The function of the cell is depicted to the right of the cell name and, if applicable, the pseudo-operation code.

MC1OUT	LOCATION OF CONTROL WORD OF OUTPUT DATA FOR OOCAPL OR OOCAPO
MC2OUT	LOCATION OF CONTROL WORD INDICATING WHERE TMOCAP BEGINS
MC3ORT	LOCATION TESTED FOR 3D ORBIT
MCABRE	OCT 20000120 LAUNCH OFF, ABORT REENTRY LIGHTS ON
MCACQ1	PZE TMLSOX, ,4
MCACQ2	ESTIMATE TIME (IN MINUTES) OF HORIZON CROSSING FOR STATION
MCACTV	ACTIVATION MASK FOR RTC SUB-CHANNELS
MCALBL	PZE TMLB1
MCALBT	PZE TMLB2
MCALM1	PZE 43
MCALM2	PZE 44
MCBETA	INDICATES ON-OR OFF-LINE STATUS OF THIS 7090
MCBTMN	DEC 162
MCBNOT	PZE 0
MCCHEK	PZE 0
MCCNTR	COUNTER USED BY PREF+SUFFIX OF R5RARF
MCCOM1	PZE 0 GEB COMPUTED DATA INDICATOR

MC 102

MCOM2 PZE 0 IP709 COMPUTED DATA INDICATOR
 MCCPNI PZE TDRANV+15 USED TO UNIQUE NOCPNI
 MCDIAG CONTAINS COUNT OF ENTRIES TO MODIAG
 MCDOWN PZE
 MCDRAG NUMERICAL INTEGRATION
 MCEFTS MASK USED TO DETERMINE CAUSE OF TRAP ON CHANNEL A
 MCESAB PZE
 MCFINI PZE 0.0 LAUNCH LOW ABORT FINISHED IF NOT=0
 MCFPTX DUMMY ON LINE MESSAGE NUMBER TO SHOW LOC. OF F.P. TRAP (MTFLPT)
 MCGTIN GREENWICH TIME OF INSERTION-FIXED PT SEC
 MCGTLO SECONDS FIXED PT MIDNITE TO LIFT OFF(GMT)
 MCHFS1 OCT 1
 MCHFSC NUMBER OF 1/2 SECONDS SINCE MID-NIGHT PRECEEDING LAUNCH
 MCHOMC DEC 5
 MCHOMS PZE HIGH SPEED OUTPUT MESSAGE CONTROL STORAGE
 MCHSOD NUMBER OF CHARACTERS TO BE SENT ON HS LINE 3
 MCHSOP NUMBER OF CHARACTERS TO BE SENT ON HS LINE 4
 MCHST1 HS GE-BURROUGHS
 MCHST2 HS 709
 MCISIN CONTAINS INTERNAL RADAR STATION NUMBER
 MCISTN BSS 1.0
 MCLED1 STATION NUMBER OF DATA BLOCK TO BE EDITED
 MCLEDD EDIT
 MCLENT DIFFERENTIAL CORRECTION
 MCLFTM PZE 0.0.0 FLT PT TIME OF LIFT-OFF
 MCLGOT PZE LAST TIME CCMAINP GOOD DATA
 MCLMSG PZE 0.0.0 CODED MESSAGE NUMBERS LAUNCH OUTPUT
 MCLNAB OCT 300
 MCLNCH OCT 200
 MCLNOR OCT 20000040
 MCLNRE OCT 220
 MCLST1 PZE 0 GEB HALF SECOND TIME FO LAST ACC MSG
 MCLST2 PZE 0 IP709 HALF SECOND TIME OF LAST ACC MSG
 MCLTMN BSS 1.F PRESENT TIME IN CCMAIN
 MCLTP1 WTBB MULTP1
 MCLTP2 WTBB MULTP2
 MCMARF PZE
 MCMINS PZE CONTAINS MIN SINCE 12 P.M. GMT
 MCMPTE NO OF WORDS NEEDED TO COMPLETE MESSAGE
 MCMSNO REQUESTED MESSAGE NUMBER(MYMESS)
 MCMST1 PZE 0 GEB FIRST DATA FRAME MACH TIME TAG
 MCMST2 PZE 0 IP709 FIRST DATA FRAME MACH TIME TAG
 MCMTPR TRAP RETURN INSTRUCTION TO PRINT ROUTINE
 MCNRRF PZE CONTAINS NO OF RETRO ROCKETS FIRED
 MCOGOD NUMBER OF PACKED WORDS FOR LINE 4
 MCOLAB NUMBER OF PACKED WORDS FOR LINE 3
 MCORRE OCT 40000020 ORBIT LIGHT OF REENTRY LIGHT ON
 MCPASN CONTAINS PASS NO RELATIVE TO CAPE C.
 MCPASS PZE 0.0.1
 MCPGMT PZE PRESENT GMT FIXED SEC B35
 MCPHSE THE PREFIX GIVES INDIC OF PHASE 00-LAU 01-ABOR 10-ORB 11-REEN
 MCPRLG LOGGING BUFFER OF ON LINE OUTPUT (LOG MACRO)
 MCRADR BERMUDA COMMUNICATION BETWEEN MPHISIN AND BIHSIN
 MCRCMD IOCT TMRST1.0+0,MNNWRB+MNNWR1+MNNWR2
 MCRECC PZE 0
 MCREEN PZE
 MCRRRS PZE
 MCRSTP PZE 1
 MCRTRD PZE 0
 MCRTMS BSS 1.0
 MCS709 PZE 0 IP709 SELECTED SOURCE INDICATOR
 MCSAVE CONTAINS FIRST LOC IN WHICH TO SAVE PANEL ON INTERRUPT
 MCSDHA PZE 0.0.0 FLAG-GOOD SELECTED DATA ARRIVED
 MCSECT NUMBER OF PRESENT SECTOR PLUS ONE. INITIALIZE AT 2

MCSELM PZE 0*0*0 BAD SELECTED DATA COUNTER LAUNCH
 MCSELS PZE 0*0*0 SELECTED SOURCE INDICATOR
 MCSEN1 TIME FOR LOGGING(MYSENS)
 MCSEN2 DCC SUB CHANNEL CONTROL(MYSENS)
 MCSGEB PZE 0 GEB SELECTED SOURCE INDICATOR
 MCSKPM LOCATIONS TESTED TO SKIP REENTRY TBL MESSAGE
 MCSSIP PZE 0*0 SELECTED SOURCE IN PROCESS
 MCTABT PZE 0*0*0 ABORT PHASE INDICATOR
 MCTDEL DELAY BEFORE ENTERING LAUNCH
 MCTEL1 PZE 0 GEB TELEMETRY DATA INDICATOR
 MCTEL2 PZE 0 IP709 TELEMETRY DATA INDICATOR
 MCTGP1 PZE 0*0 LAST T INPUT PROCESS. GOOD DATA L1
 MCTGP2 PZE 0*0 LAST T INPUT PROCESS. GOOD DATA L2
 MCTHLD PZE 0*0*0 HOLD PHASE INDICATOR
 MCTHSM BERMUDA MASK FOR H S INPUT BLOCK(MTHSIN)
 MCTHSN BERMUDA SAVED INDICATORS(MTHSIN+MPHSIN)
 MCTLST PZE 0
 MCTLTM BSS 1,F TIME TO FIND TELEMETRY
 MCTM01 BERM LOGGING CODE AND SAVED INDICATORS(MYTMIN)
 MCTM02 BERM DCC CONTROL MASK(MYTMIN)
 MCTM03 BERM LOGGING TIME(MYTMIN)
 MCTMWT TIME TO ENTER LAUNCH
 MCTOFS DEC 60180
 MCTPOS KEEPS TRACK OF MESSAGE TAPE POSITION
 MCTTIN PZE **0
 MCTTOX NUMBER OF PACKED TTY WORDS LINE 10
 MCTTOY NUMBER OF PACKED TTY WORDS LINE 11
 MCWCH2 PZE 0 IP709 DATA SOURCE INDICATOR
 MCWDCT WORD COUNT FOR MAN. INS. MESSAGE
 MCWWWV PZE 0
 MCX4RA CONTAINS INDEX REG 4 AND RETURN ADDRESS (FOR MOSAVE)
 MCZRWX PZE